

Big Data
Foundation and Management

段竹　田宏　主编

清华大学出版社

大数据基础与管理

段 竹 田 宏 主 编
吴旭东 吴 镝 冯瑞芳 朱 毅 于书皓 编 著

清华大学出版社
北 京

内 容 简 介

本书从理论结合实践的角度，讲解大数据的概念和技术。全书共分为 7 章，主要内容包括什么是大数据、大数据的特征、大数据的作用与应用、大数据的技术与分析；通过实例讲解 Data Studio 的使用方法，DB2、UDB 和 JDBC 的相关知识，集成数据管理的知识，IBM InfoSphere 软件；详细讨论大数据环境下的安全与治理；通过实例讲解了 Hadoop 技术。

图书在版编目(CIP)数据

大数据基础与管理/段竹，田宏主编. --北京：清华大学出版社，2016
ISBN 978-7-302-42523-6

Ⅰ. ①大…　Ⅱ. ①段… ②田…　Ⅲ. ①数据处理 ②数据管理　Ⅳ. ①TP274

中国版本图书馆 CIP 数据核字(2016)第 000889 号

责任编辑：刘向威
封面设计：
责任校对：焦丽丽
责任印制：

出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　　**邮　　编**：100084
社 总 机：010-62770175　　**邮　　购**：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质量反馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
课件下载：http://www.tup.com.cn，010-62795954
印 刷 者：
装 订 者：
经　　销：全国新华书店
开　　本：185mm×260mm　　**印　　张**：12.25　　**字　　数**：300 千字
版　　次：2016 年 3 月第 1 版　　**印　　次**：2016 年 3 月第1 次印刷
印　　数：1～　000
定　　价：　.00 元

产品编号：057643-01

前言 FOREWORD

大数据作为一种重要的战略资产，已经不同程度地渗透到各个行业领域和部门，其深度应用不仅有助于企业经营活动，还有利于推动国民经济发展。大数据的快速发展使它成为 IT 领域的又一大新兴产业。

大数据目前在很多的行业都有应用，包括大科学(Megascience)、RFID、感测设备网络、天文学、大气学、基因组学、生物学、大社会数据分析、互联网文件处理、制作互联网搜索引擎索引、通信记录明细、军事侦察、社交网络、通勤时间预测、医疗记录、照片图像和视频封存、大规模电子商务等。

对大数据的处理分析正在成为新一代信息技术融合应用的节点。移动互联网、物联网、社交网络、数字家庭、电子商务等是新一代信息技术的应用形态，这些应用不断产生大数据。云计算为这些海量、多样化的大数据提供存储和运算平台。通过对不同来源数据的管理、处理、分析与优化，将结果反馈到上述应用中，创造出巨大的经济和社会价值。

因此，了解大数据的概念，掌握与大数据相关的技术，对于计算机专业的学生来说是十分必要的。

本书从理论结合实践的角度，讲解大数据的概念和技术。读者从本书可以了解到什么是大数据、大数据的特征、大数据的作用与应用、大数据的技术与分析、集成数据管理、大数据环境下的安全与治理、大数据相关技术的使用等知识。

本书共分为 7 章。

第 1 章介绍什么是大数据、大数据的特征、大数据的作用与具体应用、大数据的技术与大数据的分析。

第 2 章通过实例详细讲解 Data Studio 的使用方法。

第 3 章介绍 DB2 与 JDBC 支持，讲解 DB2、UDB 和 JDBC 的通用驱动程序。

第 4 章详细讲解集成数据管理的知识，包括集成数据管理的基本知识、数据建模和设计、数据模型管理、主数据管理、元数据管理、数据的交付。

第 5 章详细讲解 IBM InfoSphere 软件。

第 6 章详细讲解大数据环境下的安全与治理，包括大数据环境下的信息安全、大数据面临的安全问题、大数据安全的应对策略、大数据的治理、大数据加密技术。

第 7 章通过实例详细讲解 Hadoop 技术。

本书不足之处恳请读者指正。

编　者

2015 年 9 月

CONTENTS 目录

第 1 章　大数据概论 …… 1
1.1　什么是大数据 …… 1
1.2　大数据的特征 …… 1
1.3　大数据的作用与具体应用 …… 2
1.3.1　大数据的作用 …… 2
1.3.2　大数据的应用 …… 3
1.4　大数据的技术与大数据的分析 …… 4
1.4.1　概述 …… 4
1.4.2　大数据的技术 …… 4
1.4.3　大数据的分析 …… 5
1.4.4　大数据现状 …… 5
1.4.5　展望大数据 …… 6
第 2 章　Data Studio …… 9
2.1　创建数据库 …… 9
2.1.1　命令方式创建数据库 …… 9
2.1.2　通过数据库向导来创建数据库 …… 9
2.2　创建数据库对象 …… 13
2.2.1　创建模式 …… 13
2.2.2　创建表空间 …… 17
2.2.3　创建缓冲池 …… 23
2.2.4　创建表 …… 25
2.2.5　创建索引 …… 31
2.2.6　创建视图 …… 37
2.2.7　创建别名 …… 40
2.2.8　创建约束 …… 40
2.2.9　创建触发器 …… 40
2.3　备份和恢复 …… 42
2.3.1　DB2 数据库的备份 …… 42
2.3.2　DB2 数据库的恢复 …… 46
2.4　DB2 优化器 …… 48
2.4.1　优化器 …… 48

2.4.2 DB2 优化器介绍 …… 49
2.4.3 SQL 语句执行过程 …… 50
2.4.4 优化器组件和工作原理 …… 52
2.4.5 扫描方式 …… 53
2.4.6 连接方法 …… 53
2.4.7 优化级别 …… 54
2.4.8 如何影响优化器来提高性能 …… 55
2.5 SQL 调优概述 …… 57
2.5.1 一般规则 …… 57
2.5.2 针对专门操作符的调优 …… 60
第 3 章 数据库开发 …… 64
3.1 DB2 与 JDBC 支持 …… 64
3.2 理解 DB2 UDB JDBC 通用驱动程序 …… 66
3.2.1 旧的 JDBC 驱动程序与新的通用 JDBC 驱动程序的比较 …… 66
3.2.2 诊断问题和分析跟踪 …… 69
3.2.3 JDBC 通用驱动程序错误代码 …… 73
第 4 章 集成数据管理 …… 76
4.1 集成数据管理简介 …… 76
4.2 数据建模和设计 …… 76
4.2.1 数据仓库设计和数据建模 …… 77
4.2.2 使用 IBM® InfoSphere® Data Architect 对 DB2 创建的全局临时表建模 …… 89
4.3 数据模型管理 …… 100
4.3.1 数据模型管理器 …… 100
4.3.2 规范 …… 101
4.4 主数据管理 …… 105
4.4.1 数据管理的范畴和主数据管理的概念 …… 106
4.4.2 主数据管理的意义 …… 107
4.4.3 主数据管理系统与数据仓库系统的关系 …… 108
4.4.4 主数据管理系统和 ODS 的关系 …… 110
4.4.5 主数据管理解决方案介绍 …… 110
4.4.6 企业主数据管理系统逻辑架构 …… 110
4.5 元数据管理 …… 112
4.5.1 明确元数据管理策略 …… 112
4.5.2 元数据集成体系结构 …… 113
4.5.3 实施元数据管理 …… 117
4.6 数据的交付 …… 119
第 5 章 IBM InfoSphere 软件 …… 121
5.1 InfoSphere Data Architect …… 121
5.1.1 什么是 IBM InfoSphere Data Architect …… 121
5.1.2 下载 DB2 Express-C …… 122

5.1.3　安装 InfoSphere Data Architect …… 122
5.2　InfoSphere Streams …… 128
5.2.1　安装流计算：一种新的计算模式 …… 129
5.2.2　InfoSphere Streams 概述 …… 129
5.2.3　流处理语言 …… 130
5.2.4　开发环境 …… 133
5.2.5　BigInsights 和 InfoSphere Streams 之间的集成和交互 …… 134
5.2.6　InfoSphere Streams 环境 …… 134
5.2.7　InfoSphere Streams 编程 …… 135
5.2.8　操作符和工具集 …… 136
5.2.9　InfoSphere Streams 集成 …… 137
5.2.10　导航信息中心 …… 137
5.3　IBM InfoSphere BigInsights …… 138
5.3.1　IBM InfoSphere BigInsights 简介 …… 138
5.3.2　IBM InfoSphere BigInsights 3.0 介绍 …… 139
5.3.3　IBM Big SQL 3.0 …… 140
5.3.4　企业集成 …… 141
5.3.5　GPFS File Place Optimizer …… 143
5.3.6　IBM Adaptive MR …… 144
5.3.7　IBM BigSheets …… 145
5.3.8　高级文本分析 …… 147
5.3.9　Solr …… 147
5.3.10　改进工作负载调度 …… 148
5.3.11　压缩 …… 149
5.3.12　总结 …… 150
第 6 章　大数据环境下的安全与治理 …… 151
6.1　大数据环境下的信息安全 …… 151
6.1.1　信息安全的发展 …… 151
6.1.2　数据安全的概念 …… 151
6.1.3　大数据的特征 …… 152
6.1.4　大数据给信息安全带来新的挑战和机遇 …… 153
6.2　大数据面临的安全威胁 …… 154
6.3　大数据安全的应对策略 …… 155
6.3.1　大数据存储安全策略 …… 155
6.3.2　大数据应用安全策略 …… 156
6.3.3　大数据管理安全策略 …… 156
6.4　大数据的治理 …… 157
6.4.1　大数据环境下的安全技术体系框架 …… 157
6.4.2　大数据治理定义 …… 157
6.4.3　数据治理的作用 …… 157

6.5 大数据加密技术 …… 158
第7章 Hadoop技术 …… 162
7.1 Hadoop简介 …… 162
7.1.1 简介 …… 162
7.1.2 Hadoop用途 …… 162
7.2 Hadoop安装与简单配置 …… 163
7.2.1 Linux安装 …… 163
7.2.2 JDK安装 …… 173
7.2.3 Hadoop下载 …… 176
7.2.4 Hadoop单机模式配置 …… 177
7.2.5 Hadoop伪分布模式配置 …… 178
附录A …… 187
参考文献 …… 188



第1章

大数据概论

1.1 什么是大数据

大数据(Big data),或称巨量数据、海量数据,指的是所涉及的数据量的规模巨大到无法通过人工,在合理时间内达到截取、管理、处理、并整理成为人类所能解读的信息。

1.2 大数据的特征

目前较为普遍的大数据定义为"大数据主要指无法使用传统流程或工具处理和分析的数据"。IBM 所称的大数据通常用 3 个特征描述:数量(Volume)、种类(Variety)和速度(Velocity)。2012 年,英特尔大数据论坛上,IDC 定义了大数据的四大特征:海量的数据规模;快速的数据流转和动态的数据体系;多样的数据类型;巨大的数据价值。综合上述定义,可以用大量(Volume)、多样性(Variety)、速度快(Velocity)以及价值高和密度低(High Value and Low Density)四大特征来描述大数据。

(1) 大量化(Volume)。数据量级已从 GB 至 TB、PB 乃至 ZB 上升,可称海量、巨量甚至超量,且仍在持续爆炸式增长。据 WinterCorp 调查显示,最大数据仓库中的数据量,每两年增加 3 倍左右,其增长速度远超摩尔定律增长速度。谷歌公司每天要处理超过 24PB 的数据,Facebook 每天更新的照片量超过 1000 万张,Twitter 每天都会发布超过 4 亿条微博。截止到 2012 年 12 月底,中国网页数量为 1227 亿个左右,比 2011 年同期增长 41.7%左右。大数据环境下,网络信息的规模急剧增长,PB 级甚至 ZB 级的数据需要大规模并行计算网络的支持,巨大的存储、链接、传输和加密归并等开销使常规加密计算不堪重负。

(2) 多样化(Variety)。数字信息由原来简单的数值、字符和文本向网页、图片、视频、图像和位置信息等半结构化和非结构化的数据类型发展,并且信息大多分布在不同的地理位置、不同的存储设备以及不同的数据管理平台。此外,互联网环境中大量的信息缺乏有效的组织,信息的无序化大幅度降低了查找和利用信息的效率,阻碍了有价值信息的加密和管理效率。具体体现在如下 3 个方面:

- 数据来源多。随着互联网和物联网技术的飞速发展,可以通过微博、社交网站、电子商务网站、车载网以及遍布全球的各式各样的传感器等多种数据来源获取数据。
- 数据类型繁多。传统数据大多以表格的形式保存,而大数据中70%~85%的数据是图片、音频、视频、网络日志、链接信息等半结构化和非结构化的数据。
- 数据之间关联性强,交互频繁。如大型的电子商务网站和社交网络中,一些用户的点击行为在一定程度上反映了该用户潜在的兴趣爱好和需求,链接之间的关联性较强。

(3) 快速化(Velocity)。大数据的时效性要求对数据的处理能够做到实时、快速,要达到这一目标,要求使用的硬件平台亦能够同步更新换代,并将分布式计算、并行计算、软件工程、人工智能等技术应用到其中。

(4) 价值高和密度低(High Value and Low Density)。互联网充斥着大量重复和虚假信息,通常有价值的信息较为分散,密度很低。

正是大数据的价值具备稀疏性、多样性和不确定性的特点,较多数据采集和存储系统又要求能够快速访问大数据的历史版本数据,备份数据的保存期限更长,备份的窗口不断缩短,很多数据需要在线备份和故障实时恢复等,大数据的安全维护对存储资源、计算资源、网络资源等都提出了极高的性能需求,其安全存储与数据保护面临着前所未有的压力和挑战。

1.3 大数据的作用与具体应用

1.3.1 大数据的作用

(1) 对大数据的处理分析正在成为新一代信息技术融合应用的节点。移动互联网、物联网、社交网络、数字家庭、电子商务等是新一代信息技术的应用形态,这些应用不断产生大数据。云计算为这些海量、多样化的大数据提供存储和运算平台。通过对不同来源数据的管理、处理、分析与优化,将结果反馈到上述应用中,将创造出巨大的经济和社会价值。大数据具有催生社会变革的能量。但释放这种能量,需要严谨的数据治理、富有洞见的数据分析和激发管理创新的环境(Ramayya Krishnan,卡内基·梅隆大学海因兹学院院长)。

(2) 大数据是信息产业持续高速增长的新引擎。面向大数据市场的新技术、新产品、新服务、新业态会不断涌现。在硬件与集成设备领域,大数据将对芯片、存储产业产生重要影响,还将催生一体化数据存储处理服务器、内存计算等市场。在软件与服务领域,大数据将引发数据快速处理分析、数据挖掘技术和软件产品的发展。

(3) 大数据利用将成为提高核心竞争力的关键因素。各行各业的决策正在从"业务驱动"转变为"数据驱动"。

对大数据的分析可以使零售商实时掌握市场动态并迅速做出应对;可以为商家制定更加精准有效的营销策略提供决策支持;可以帮助企业为消费者提供更加及时和个性化的服务;在医疗领域,可提高诊断准确性和药物有效性;在公共事业领域,大数据也开始发挥促进经济发展、维护社会稳定等方面的重要作用。

(4) 大数据时代科学研究的方法手段将发生重大改变。例如,抽样调查是社会科学的基本研究方法。在大数据时代,可通过实时监测、跟踪研究对象在互联网上产生的海量行为

数据，进行挖掘分析，揭示出规律性的东西，提出研究结论和对策。

1.3.2　大数据的应用

大数据目前在很多的行业都有应用，包括了大科学(Mega Science)、RFID、感测设备网络、天文学、大气学、基因组学、生物学、大社会数据分析、互联网文件处理、制作互联网搜索引擎索引、通信记录明细、军事侦察、社交网络、通勤时间预测、医疗记录、照片图像和视频封存、大规模的电子商务等。下面将给出几个具体的大数据应用案例供大家参考。

(1) 大科学

大科学领域的一个典型代表是大型强子对撞机(Large Hadron Collider)，大数据在大科学中有着广泛的应用。图 1.1 所示为大型强子对撞机的一部分。大型强子对撞机中有 1.5 亿个感测器，每秒发送 4000 万次的数据。实验中每秒产生将近 6 亿次的对撞，在过滤去除 99.999%的撞击数据后，得到约 100 次的有用撞击数据。将撞击结果数据过滤处理后仅记录了 0.001%的有用数据，四个对撞机的全部数据量复制前每年产生 25 拍字节(PB)，复制后为 200 拍字节。

如果将所有实验中的数据在不过滤的情况下全部记录，数据量将会变得过度庞大且很难处理。每年数据量在复制前估计将会达到 1.5 亿拍字节，等于每天有近 500 艾字节(EB)的数据量。这个数字代表每天实验将产生相当于 500 垓(5×10^{20})字节的数据，是全世界所有数据源总和的 200 倍左右，在如此庞大的数据中去寻找希格斯玻色子(Higgs boson)存在的证据，就需要借助大数据的应用了。



图 1.1　大型强子对撞机的一部分

(2) 社会学

大数据产生的背景离不开微博、Facebook 等社交网络的兴起，人们每天通过这种自媒体传播信息或者沟通交流，由此产生的信息被网络记录下来，社会学家可以在这些数据的基础上分析人类的行为模式、交往方式等。美国的涂尔干计划就是依据个人在社交网络上的数据分析其自杀倾向，该计划从美军退役士兵中拣选受试者，透过 Facebook 的行动应用收集资料，并将用户的活动数据传送到一个医疗资料库。收集完成的数据会接受人工智能系统分析，接着利用预测程序来即时监视受测者是否出现一般认为具有伤害性的行为。

(3) 营销与销售

企业与客户之间的关系早在第一次销售之前就已建立。大数据营销技术可以帮助企业在客户仍然研究自己的选择时与其客户接触,并证明能够提供客户需要的个性化产品和服务。如图 1.2 所示,个人感觉和真实数据的结论往往不同。

如果要赢得客户,需要单独了解客户。需要充分利用企业内外的所有数据,对客户的需求和行为进行智能预测,提高客户保留率和忠诚度。通过更好地了解客户,企业可以提供适当的服务水平,从而提高客户保留率,将客户转变为支持者。由此可见大数据可以帮助企业在营销与销售中取得优秀的成果。

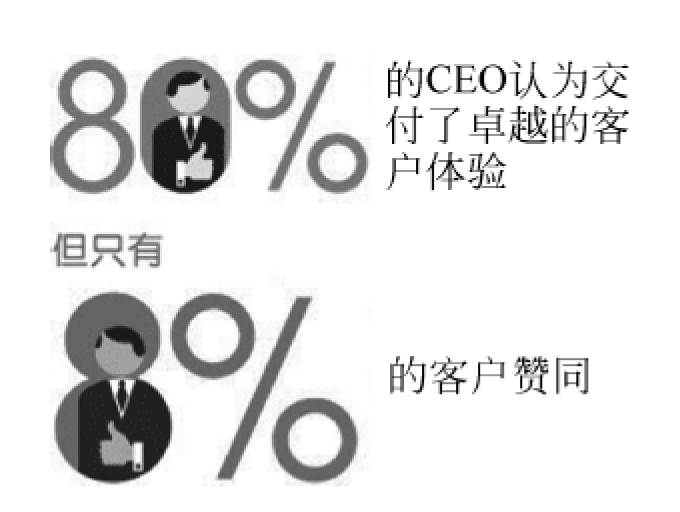


图 1.2 个人感觉和真实数据的结论往往是不同的

1.4 大数据的技术与大数据的分析

1.4.1 概述

在不同的行业领域当中,对大数据的具体运用和研究范围是不同的,普通的行业用户,例如金融、商业、遥感等领域的用户仅仅需要在应用层对大数据进行应用。而计算技术的研究人员和开发人员要对大数据技术在算法层、系统层和平台层的技术进行研究和开发。

大数据不单是单纯的事实存在的很大数量的数据。在大数据研究中,一个很重要的问题是如何对这些海量的数据进行分析,并从中得到一些有意义、有价值的数据。大数据的数据体量巨大、数据种类多样、价值密度低和要求处理速度快的四个特点表现了大数据分析不断增加的复杂性。

本节将对大数据的技术和大数据的分析进行简要的介绍。

1.4.2 大数据的技术

不同的用户对大数据技术的运用是不同的,对大数据的研究领域也不尽相同。本节将对不同层次的大数据技术进行介绍。

大数据的最顶层为应用层,这一层的研究层次包含大数据行业应用/服务层与应用开发层。在这个层次上,大数据需要的技术有行业应用系统和服务的使用与开发,还有领域应用、服务需求和计算模型的制定技术等。

第二层为算法层,包括应用算法层与基础算法层。

在应用算法层,大数据主要的研究内容为社会网络,商业智能,推荐系统,自然语言处理与 Web 数据的挖掘与搜索等。在每个研究内容中又会细分为很多项具体技术,由于篇幅限

制，在这里不多做介绍。

在基础算法层，对于大数据的主要研究内容为大数据并行化机器学习和挖掘方法。具体的算法包括分类算法（Classification）、聚类（Clustering）和参数估计（Parameters Estimation）等。

第三层为系统层，该层包括了并行编程模型与计算框架层和大数据存储管理层。

在并行编程模型与计算框架层，研究领域包括了 MapReduce 在不同构架上实现的技术，定制式并行计算框架与 CUDA、MPI 等技术。

在大数据存储管理层，大数据的研究领域包括大数据预处理技术，记录型大数据索引和查询技术，SQL/NoSQL 查询语言接口技术，分布式数据库与分布式文件系统技术等。

最底层为平台层，该层只有一层内容即并行构架和资源平台层。该层的技术主要有共享内存构架技术、分布内存构架技术、混合式构架技术与大数据应用/服务云计算支撑平台技术等。

1.4.3　大数据的分析

在大数据的研究中，如何从海量的数据中分析并获取到有价值的数据是一个非常重要的问题。大数据分析主要有五个方面，本节将介绍一些大数据分析的方法。

(1) 可视化分析。大数据分析的使用者有大数据分析专家，同时还有普通用户，但是他们二者对于大数据分析最基本的要求就是可视化分析。因为可视化分析能够直观地呈现大数据的特点，同时能够非常容易被用户所接受，就如同看图说话一样简单明了。

(2) 数据挖掘算法。大数据分析的理论核心就是数据挖掘算法。各种数据挖掘的算法基于不同的数据类型和格式才能更加科学地呈现出数据本身具备的特点，也正是因为这些被全世界统计学家所公认的各种统计方法才能深入数据内部，挖掘出公认的价值。另外一个方面也是因为有这些数据挖掘的算法才能更快速地处理大数据，如果一个算法要花上好几年才能得出结论，那大数据的价值也就无从说起了。

(3) 预测性分析。大数据分析的应用领域之一就是预测性分析，从大数据中挖掘出特点，通过科学建模之后便可以通过模型带入新的数据，从而预测未来的数据。

(4) 语义引擎。非结构化数据的多元化给数据分析带来新的挑战，这就需要一套工具能够系统地分析和提炼数据。语义引擎需要有足够的人工智能系统的功能以便从数据中主动地提取信息。

(5) 数据质量和数据管理。大数据分析离不开数据质量和数据管理，高质量的数据和有效的数据管理，无论是在学术研究还是在商业应用领域，都能够保证分析结果的真实和有价值。

总之，以上五个方面是大数据分析的基础，如果想对大数据进行更加深入的分析，那么需要一些更加独特和专业的方法对数据进行分析。

1.4.4　大数据现状

据一则大数据发展分析报告称，大数据的快速发展，使它成为 IT 领域的又一大新兴产业。据中央财经大学中国经济管理研究院博士张永力估算，国外大数据行业约有 1000 亿美元的市场，而且每年都以大约 10％的速度在增长，增速是软件行业的两倍。

1. 政府积极介入推动

2009年,联合国启动“全球脉动计划”,借大数据推动落后地区发展。2012年1月,世界经济论坛年会把“大数据、大影响”作为重要议题。美国从开放政府数据、开展关键技术研究和推动大数据应用三方面布局大数据产业。美国在开放政府上非常积极,通过Data.gov开放37万个数据集,并开放网站的API和源代码,提供数千个数据应用。除了推动本国政府数据开放外,美国还倡导发起了全球开放政府数据运动,已有41个国家响应。美国政府还投资两亿美元促进大数据核心技术的研究和应用,把大数据与集成电路、互联网放在同等重要的位置,从国家层面推进。

2. 资本市场也对大数据钟爱有加

2012年4月,大数据分析公司Splunk高调宣传大数据,引发投资者关注。12月初,为企业市场提供Hadoop解决方案的创业公司Cloudera获得6500万美元融资,估值约为7亿美元。近期,高盛联席主席斯科特·斯坦福说:“投资大数据及其运用回报率最高”。大数据领域的企业并购热度也在上升,单笔平均并购金额方面,大数据超过云计算位居IT领域榜首,在总并购额上也位居第二。

3. 人才需求巨大

据盖特纳咨询公司预测,大数据将为全球带来大约440万个IT新岗位和上千万个非IT岗位。麦肯锡公司预测美国到2018年需要深度数据分析人才44万~49万人,缺口14万~19万人;需要既熟悉本单位需求又了解大数据技术与应用的管理者约150万,这方面的人才缺口更大。中国是人才大国,但能理解与应用大数据的创新人才目前还很稀缺。

4. 国内情况

大数据的火爆,也带动了国内学术界、产业界和政府对大数据的热情。2011年以来,中国计算机学会、中国通信学会先后成立了大数据委员会,研究大数据中的科学与工程问题,科技部的《中国云科技发展“十二五”专项规划》和工信部的《物联网“十二五”发展规划》等都把大数据技术作为一项重点予以支持。其中工信部发布的物联网“十二五”规划上,把信息处理技术作为4项关键技术创新工程之一被提出来,其中包括了海量数据存储、数据挖掘、图像视频智能分析,这都是大数据的重要组成部分。而另外3项关键技术创新工程,包括信息感知技术、信息传输技术、信息安全技术,也都与“大数据”密切相关。

大数据的热潮触发了一场思想启蒙运动,使得“大数据是资产,不是包袱”、“要拿数据说话”等观念逐步深入人心,改变了以往不重视数据积累,不相信数据分析等认识。有了这种思维模式的改变,大数据的应用就有了希望。

1.4.5 展望大数据

1. 大数据推动信息产业创新

大数据是指一般的软件工具难以捕捉、管理和分析的大容量数据,一般以“太字节”为单位,大数据之“大”,并不仅仅在于“容量之大”,更大的意义在于:通过对海量数据的交换、整合和分析,发现新的知识,创造新的价值,带来“大知识”、“大科技”、“大利润”和“大发展”。信息管理专家涂子沛在其专著中如是定义大数据。

根据 IDC(国际数据公司)的监测统计,2011 年全球数据总量已经达到约 1.8ZB,而这个数值还在以每两年翻一番的速度增长,预计到 2020 年全球将总共拥有 35ZB 的数据量,增长近 20 倍。

2. 大数据将改变经济社会管理面貌

大数据作为一种重要的战略资产,已经不同程度地渗透到每个行业领域和部门,其深度应用不仅有助于企业经营活动,还有利于推动国民经济发展。麦肯锡研究表明,在医疗、零售和制造业,大数据可以每年提高劳动生产率 0.5~1 个百分点。

大数据技术作为一种重要的信息技术,对于提高安全保障能力、应急能力,优化公共事业服务,提高社会管理水平,作用正在日益凸显。大数据技术还可增强安全保障能力。在国防、反恐、安全等领域应用大数据技术,能够对来自于多种渠道的信息快速进行自动分类、整理、分析和反馈,有效解决情报、监视和侦察系统不足等问题,提高国家安全保障能力。

3. 大数据存储管理挑战及管理技术

目前电信、金融、零售等行业希望通过大数据的分析手段来帮助自己做出理性的决策。特别是电信和金融行业表现得尤为突出,市场数据没有与用户消费数据打通。面临的第一个问题就是海量数据存储的问题。多数企业正在试图建设自己的数据中心来满足大规模数据量的产生,但是随着数据的进一步增多,很多数据的查询和分析性能急剧下降,有的数据中心甚至出现了无法响应的状况,为企业的业务带来了很大损失。

企业的 CIO 们有着这样的疑虑,什么样的数据管理策略才能够对数据进行有效的保护,而且在需要时能让数据变得有价值。只有数据与适合的存储系统相匹配,制定出管理数据的战略,才能高成本、高可靠、高效益地应对大量数据。对于企业来说,大数据首先需要解决的问题就是成本和时间效应问题。商机不容错过,存储数据管理可以通过自动化操作实现,备份和归档软件可让企业的关键数据分存在不同的区域,然后按照特定的业务需求,对数据进行提取、操作和分析,并形成企业所需要的目标数据。大数据面临的存储难题迎刃而解。

大数据的关注度在不断升温,而大数据管理的技术也层出不穷。在众多技术中,有 6 种数据管理技术普遍被关注,即分布式存储与计算、内存数据库技术、列式数据库技术、云数据库、NoSQL、移动数据库技术。其中分布式存储与计算受关注度最高。

分布式存储与计算架构可以让大量数据以一种可靠、高效、可伸缩的方式进行处理。因为以并行的方式工作,所以数据处理速度相对较快,且成本较低,Hadoop 和 NoSQL 都属于分布式存储技术的范畴。

内存数据库技术可以作为单独的数据库使用,还能为应用程序提供即时的响应和高吞吐量,SAP 的 HANA 是该技术的典型代表。

列式数据库的特点是可以更好地应对海量关系数据中列的查询,占用更少的存储空间,这也是构建数据仓库的理想架构之一。

云数据库可以不受任何部署环境的影响,随意地进行拓展,进而为客户提供适宜其需求的虚拟容量,并实现自助式资源调配和自助式使用计量。目前微软公司的 SQL Server 可以提供类似的服务。

NoSQL 数据库适合于以下场景,即庞大的数据量、极端的查询量和模式演化。企业可

以通过 NoSQL 获得高可扩展性、高可用性、低成本、可预见的弹性和架构灵活性的优势，甲骨文在 2011 年推出 Oracle NoSQL 数据库。

移动数据库技术是适应移动计算的产物。随着智能移动终端的普及，人们对移动数据实时处理和管理的要求不断提高，移动数据库具有平台的移动性、频繁断接性、网络条件的多样性、网络通信的非对称性、系统的高伸缩性和低可靠性以及电源能力的有限性等，也正是因为这些特性被业界所重视。

4. 我国大数据发展策略

中科院计算所网络数据科学与工程研究中心主任程学旗在接受采访时表示：“数据的规模如此之大，现有的 IT 技术根本没有办法分析处理，价值难以得到有效利用。对这些数据的感知、分析，同时加以商业化，就是大数据技术需要完成的工作。”如何挖掘大数据的价值是重中之重。

我国应将大数据作为新一轮科技竞争和产业竞争的战略重点和制高点，充分认识“数据、技术、应用”三位一体、有机统一的内涵，掌握未来大数据发展主动权。



第2章

Data Studio

本章将会介绍 Data Studio 相关操作内容，使用软件环境为 IBM DB2 10.5 版本。

2.1 创建数据库

2.1.1 命令方式创建数据库

本小节将会介绍使用命令方式创建数据库。

CREATE DATABASE 用于创建数据库。基本语法是：

```
CREATE DATABASE database_name;
```

当然，此命令还包括许多参数选项，这里不进行详细说明。

例如：

```
CREATE DATABASE MYDB AUTHMATIC STORAGE YES ON 'C:\' DBPATH ON 'C:\'
USING CODESET GBK TERRITORY CN COLLATE USING SYSTEM PAGE SIZE 4096;
```

此命令创建了一个名为 MYDB 的数据库。

创建一个名为 dbsample 的数据库，如图 2.1 所示。

2.1.2 通过数据库向导来创建数据库

创建数据库向导可以帮助用户创建新的数据库和调整现有的数据库。操作步骤如下。

(1) 打开 IBM Data Studio 并切换到【管理资源管理器视图】，如图 2.2、图 2.3 所示。

(2) 在【管理资源管理器】对话框中右击选择【DB2】|【新建数据库】命令，如图 2.4 所示。

(3) 输入相关认证信息，单击【完成】按钮进入数据库创建向导，如图 2.5 所示。

(4) 在【详细信息】对话框中，为新数据库指定信息，如图 2.6 所示。在【存储器】对话框中，指定存储数据的位置，如图 2.7 所示。

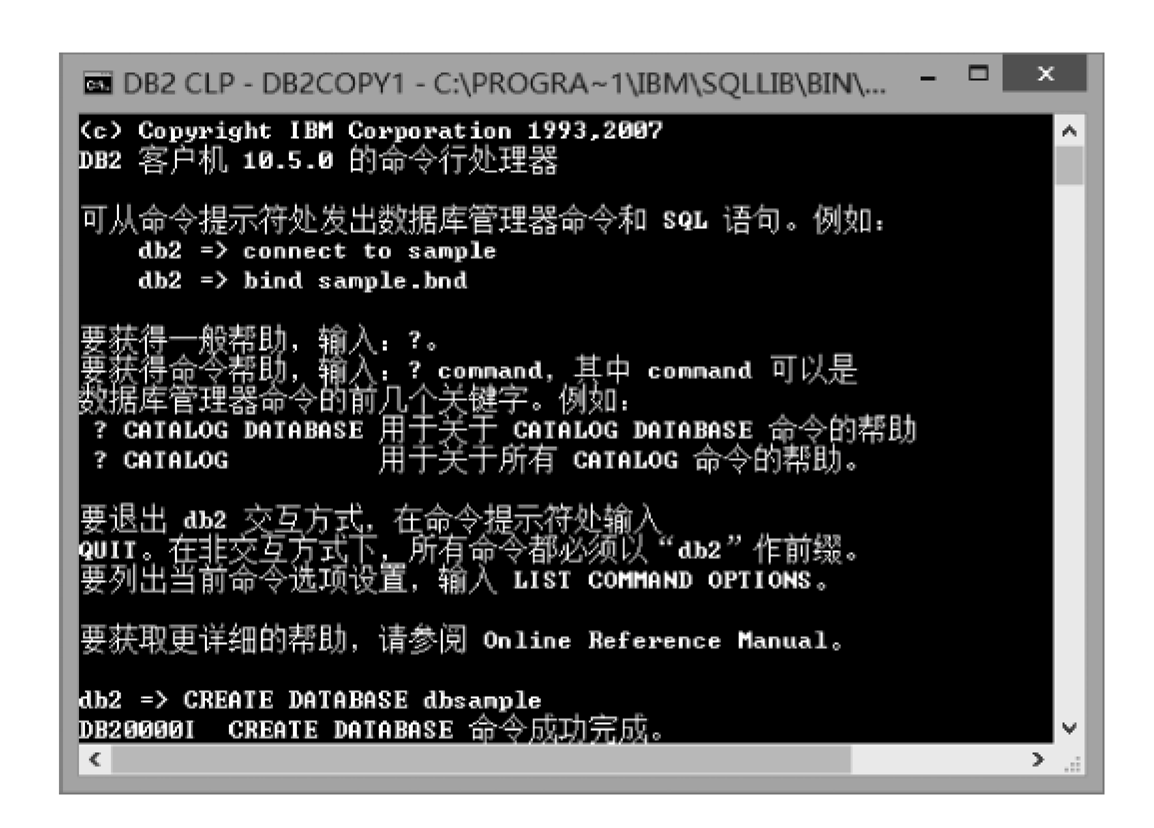


图 2.1　命令方式创建数据库



图 2.2　选择工作空间



图 2.3　切换至【管理资源管理器】



图 2.4　新建数据库



图 2.5　数据库认证信息输入

新建数据库

设置

指定要使用的任何其他设置。完成后请单击"运行"。

预览命令 运行

1. 详细信息
2. 存储器
3. 语言环境

为新数据库指定信息

输入数据库的名称，并根据自己的需求定制数据库。对于 Windows：如果 DB2_CREATE_DB_ON_PATH 未设置或设置为 NO，那么数据库位置只能是驱动器盘符。

数据库名称： MYDB

数据库位置：

C:\ 浏览...

别名：

注释：

将访问限于系统目录

缺省的缓冲池和表空间页面大小： 缺省值 (4KB)

缺省扩展数据块大小 (1-256)：

缺省表空间中的分段数 (1-256)：

命令

消息

图 2.6 为新数据库指定信息

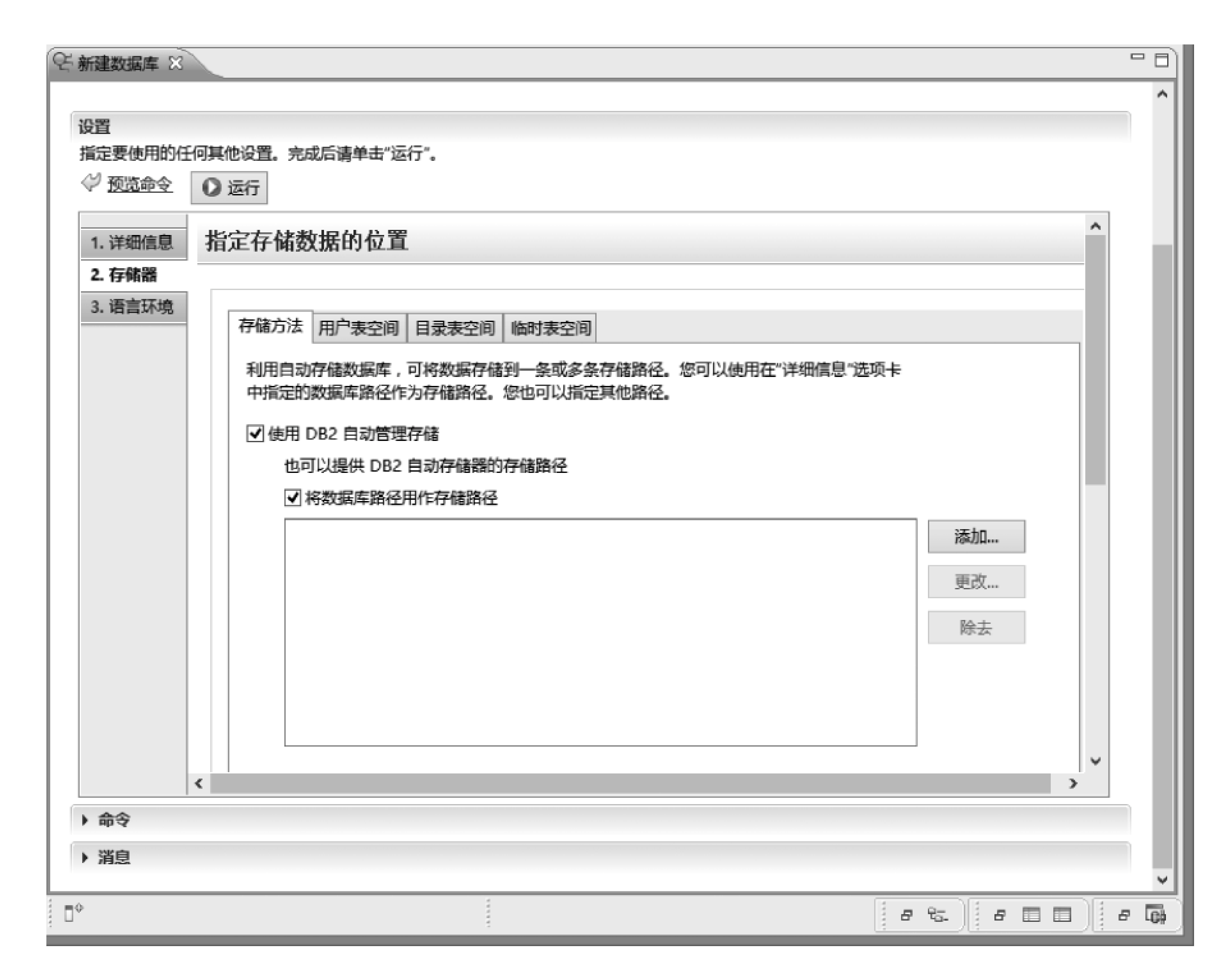


图 2.7 指定存储数据的位置

（5）指定语言环境（地域或代码集）确定数据库所使用的字符集，同时也可以确定如何比较不同的字符串。这里用默认设置，如图 2.8 所示。



图 2.8　为该数据库指定语言环境

最后，在设置无误后，单击【运行】按钮，运行创建数据库命令。

2.2　创建数据库对象

2.2.1　创建模式

（1）右击选择【所有数据库】| localhost | DB2 | MYDB |【模式】|【创建模式】命令，如图 2.9 所示。

（2）在模式的【名称】一栏中输入模式名称，如图 2.10 所示。

（3）进入【特权】选项卡中，对特权进行设置，如图 2.11 所示。

（4）进入【权限】选项卡中，对权限进行设置，如图 2.12 所示。

（5）进入【文档】选项卡中，输入文档信息，如图 2.13 所示。

（6）完成以上步骤后，新创建的模式将会出现在模式界面中，如图 2.14 所示。

（7）完成后选择【查看和部署更改】命令，如图 2.15 所示。

（8）单击【运行】命令，成功创建模式，如图 2.16 所示。

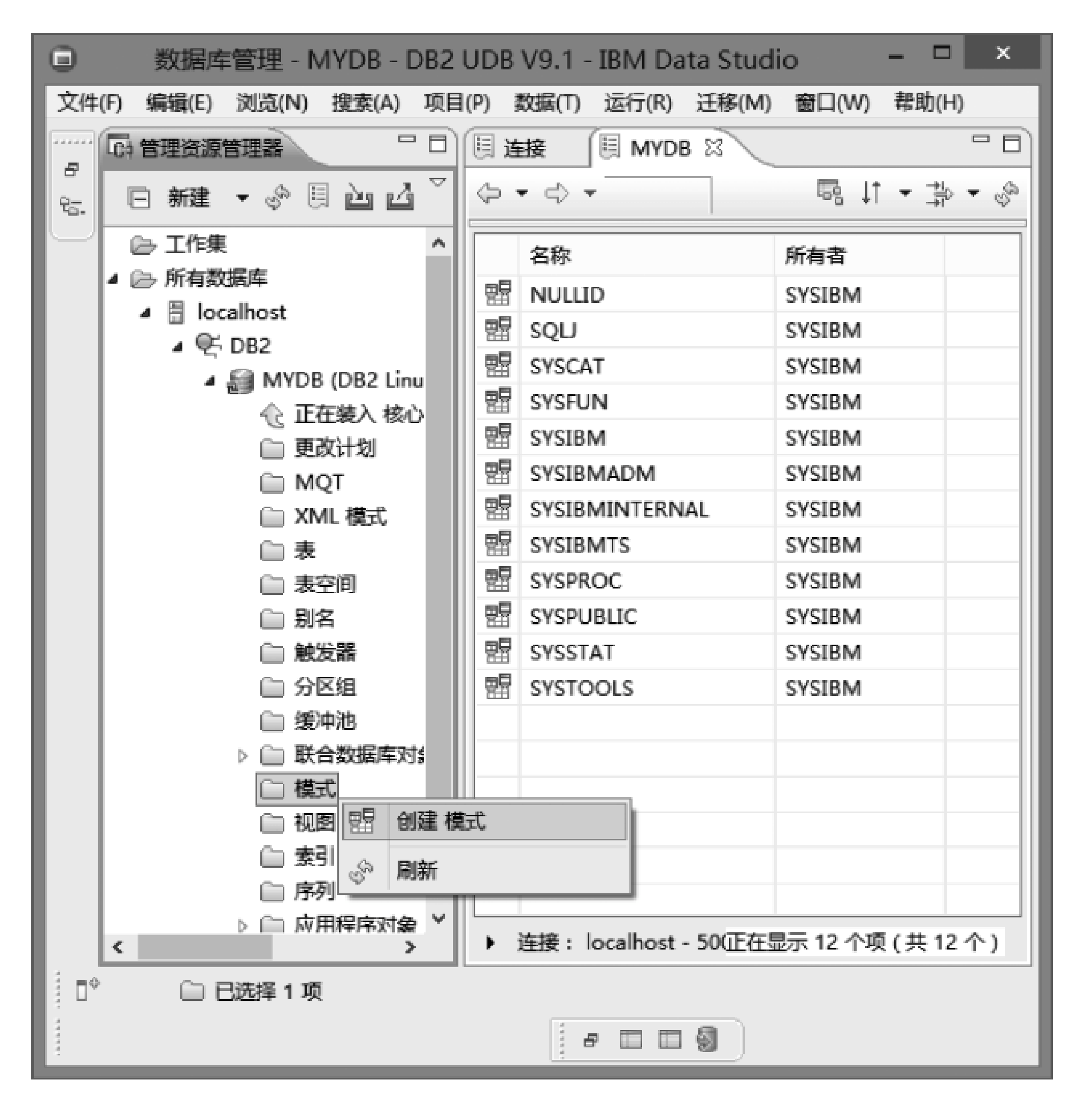


图 2.9 创建模式

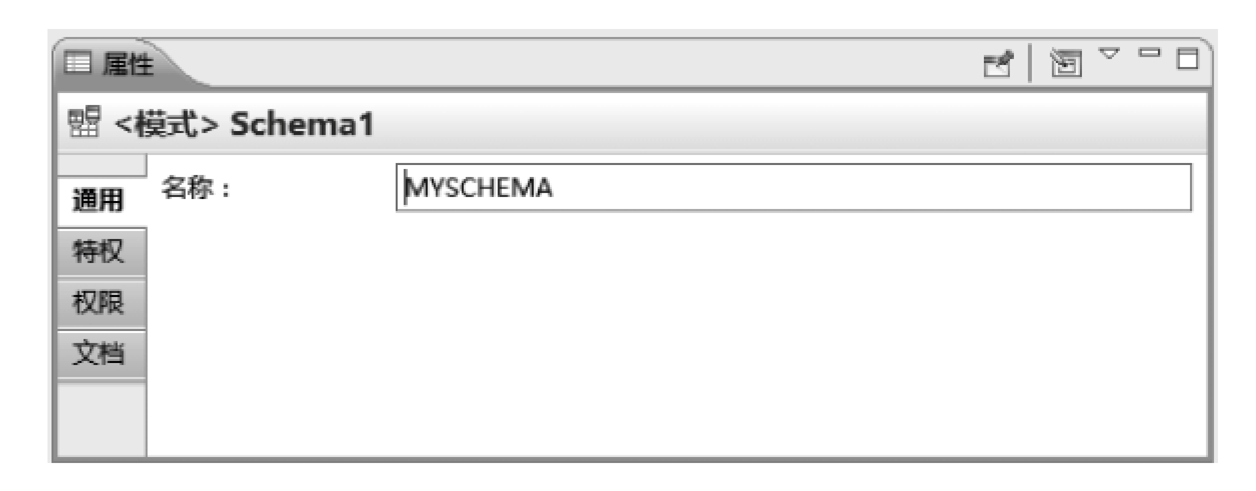


图 2.10 输入模式名称

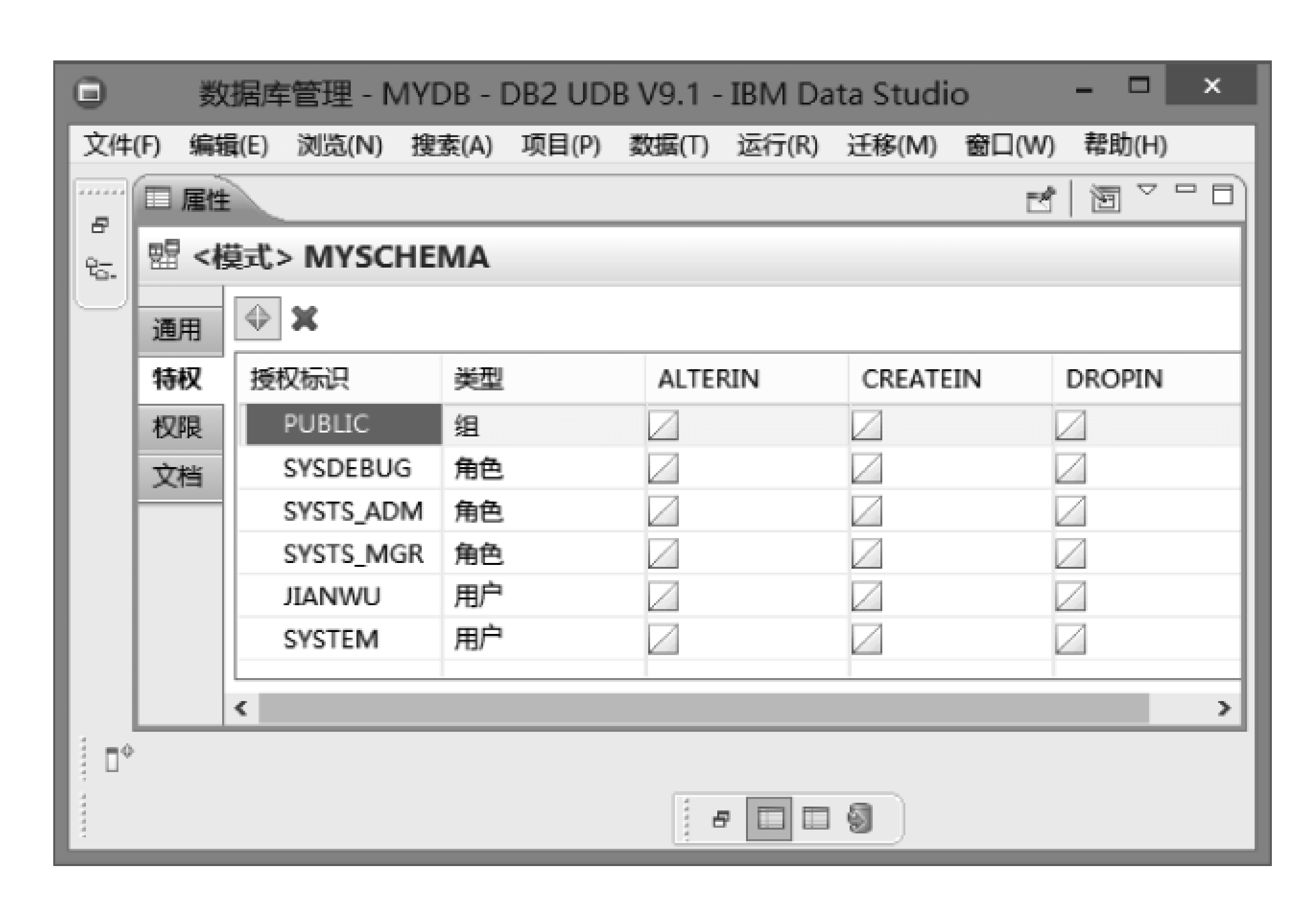


图 2.11 特权设置



图 2.12　权限设置



图 2.13　文档设置

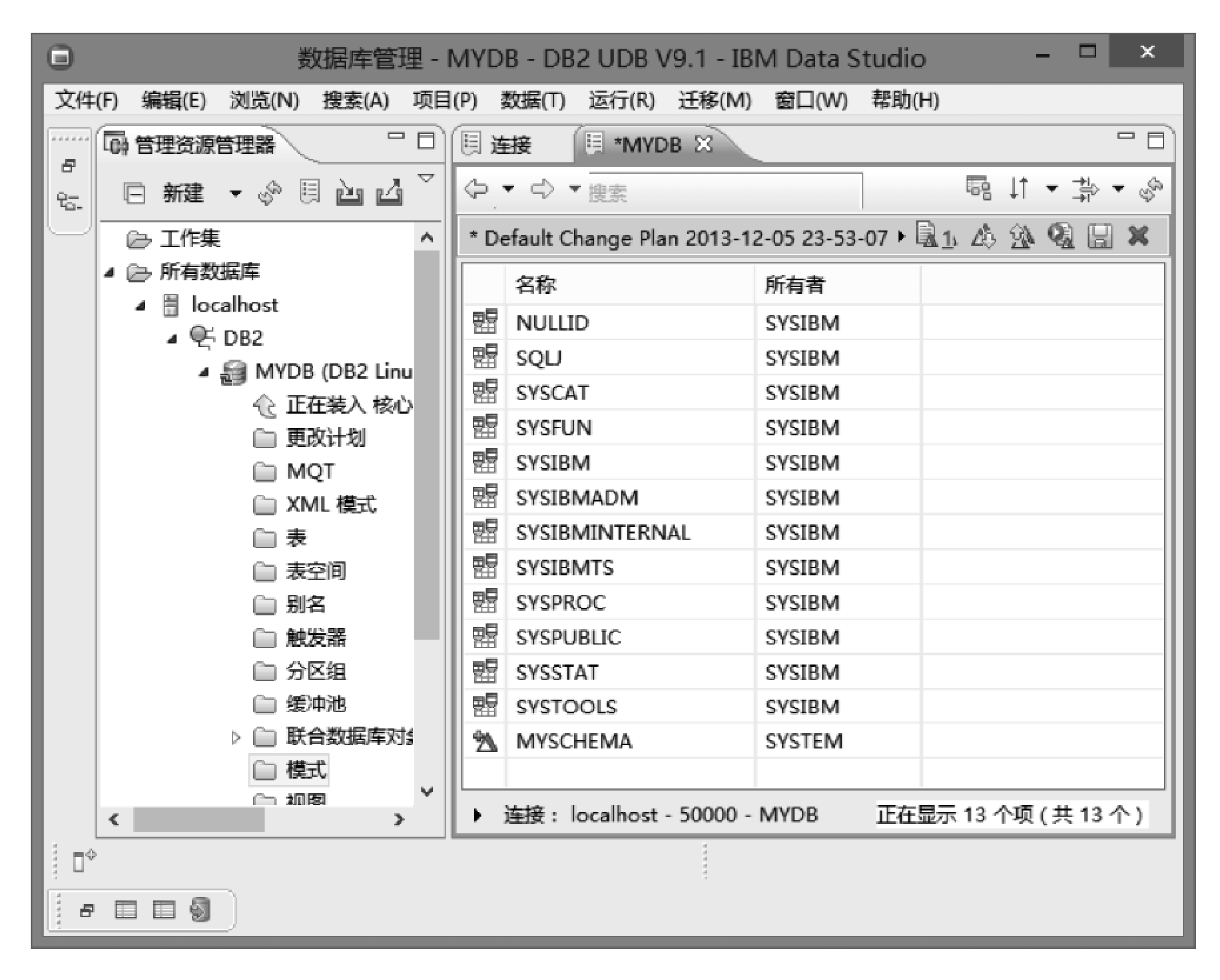


图 2.14 模式界面

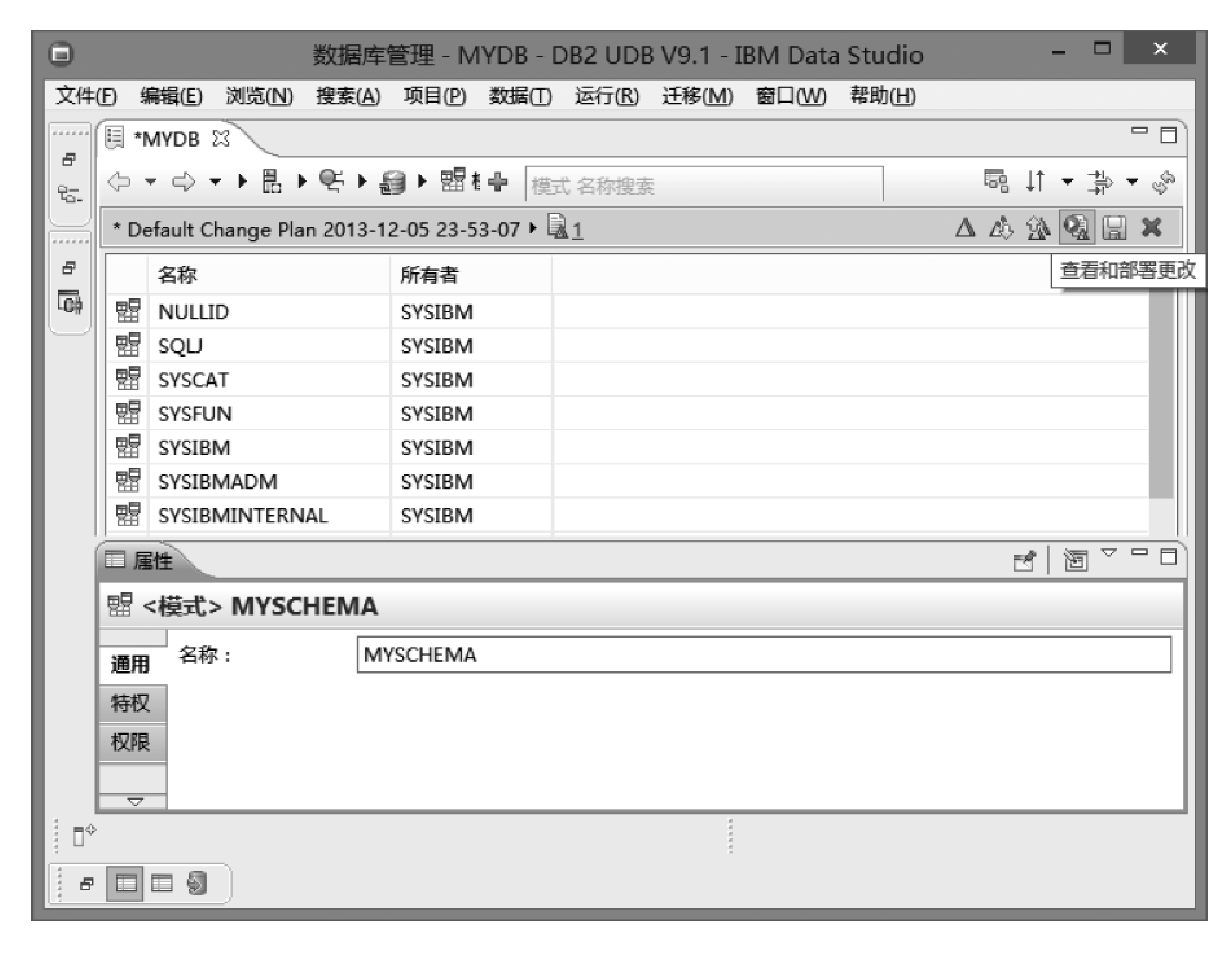


图 2.15 部署模式



图 2.16 查看命令和选择部署选项

2.2.2 创建表空间

(1) 右击选择【所有数据库】|localhost|DB2|MYDB|【表空间】|【创建常规表空间】命令，如图 2.17 所示。

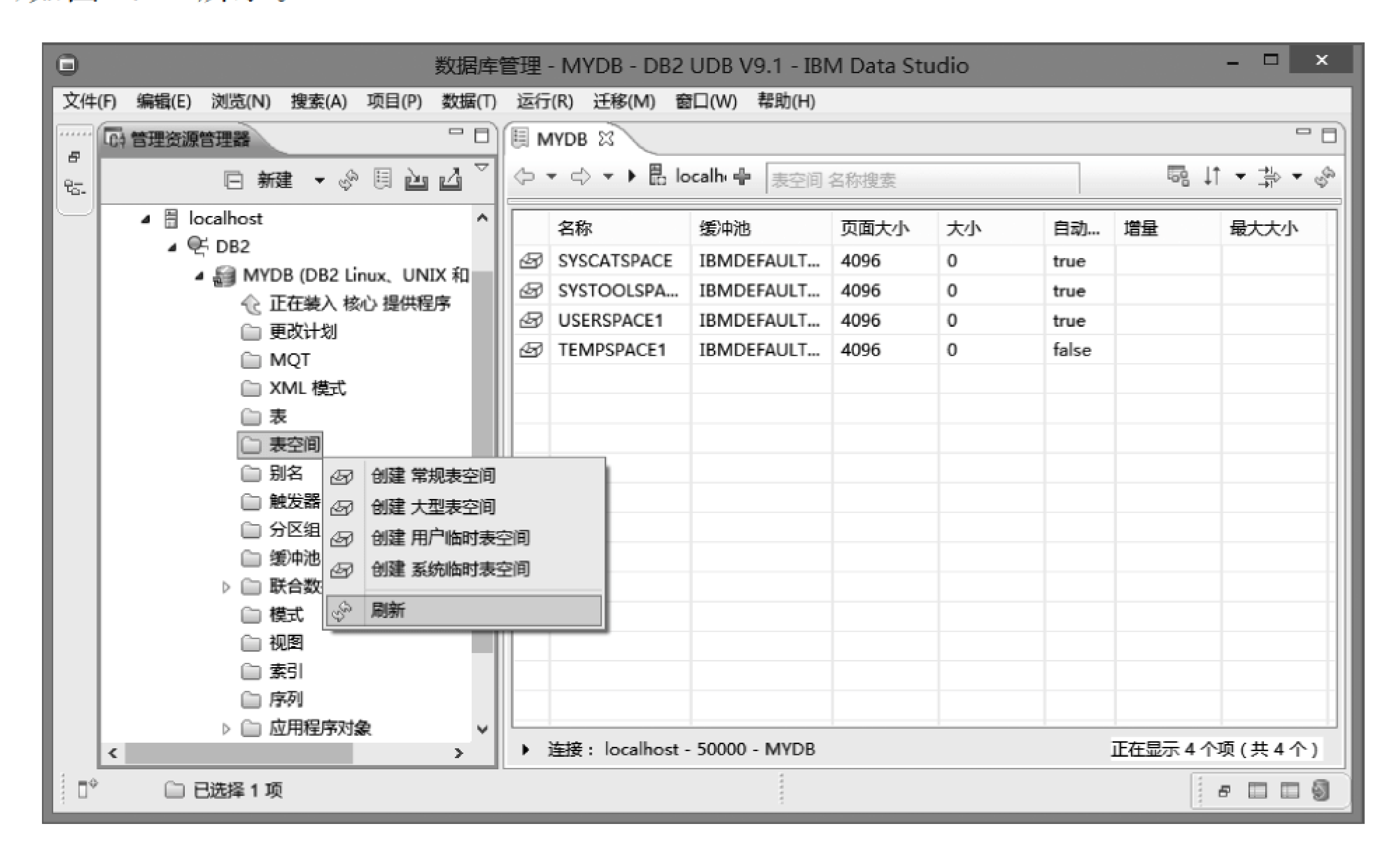


图 2.17 创建常规表空间

(2) 输入表空间的名称，并设置“管理”和“恢复”选项，如图 2.18 所示。

(3) 对表空间的【特权】进行设置，如图 2.19 所示。

(4) 对表空间的【表】进行设置，如图 2.20 所示。

(5) 对表空间的【容器】进行设置，如图 2.21 所示。



图 2.18 表空间【通用】设置

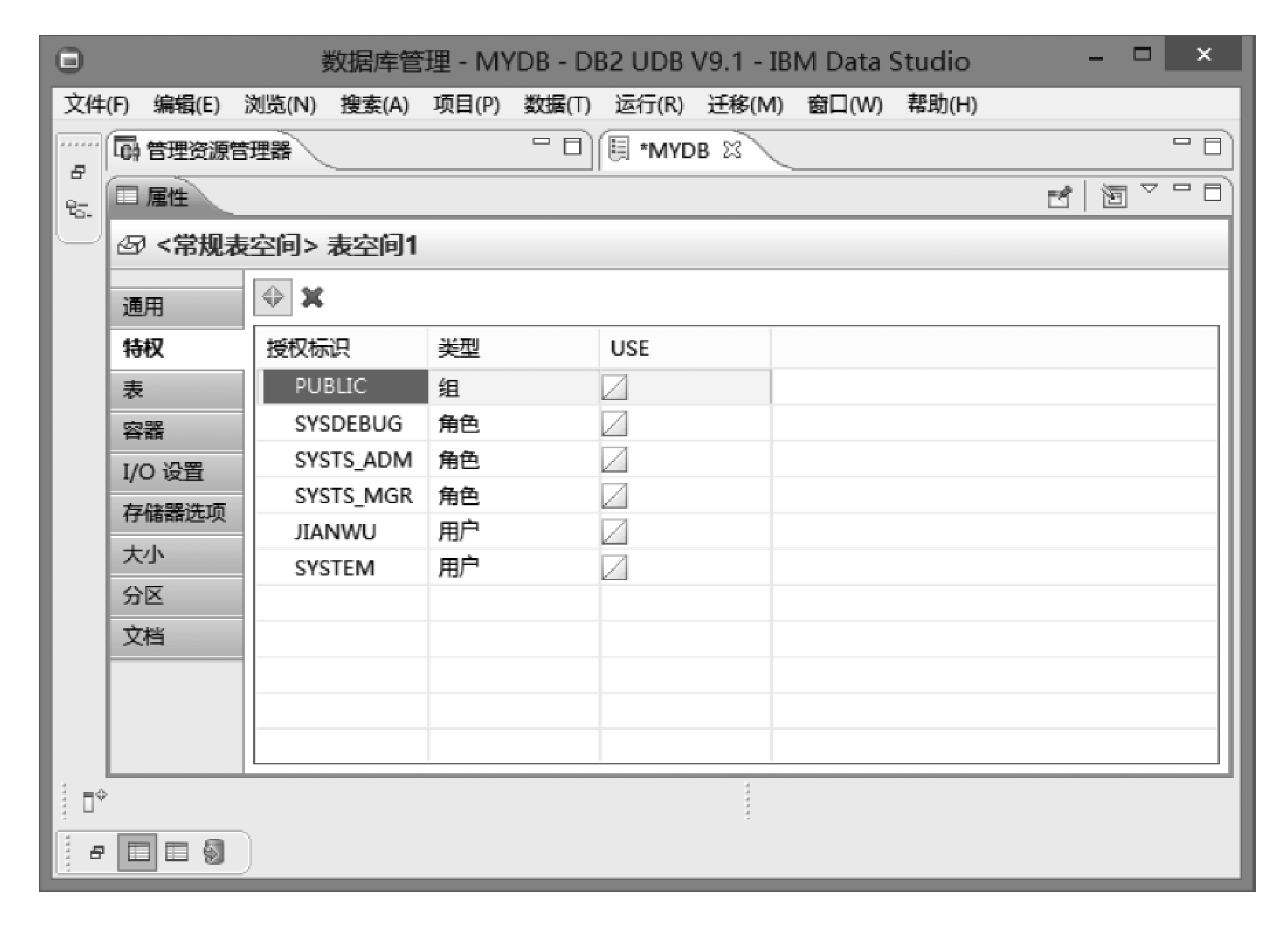


图 2.19 表空间的【特权】设置

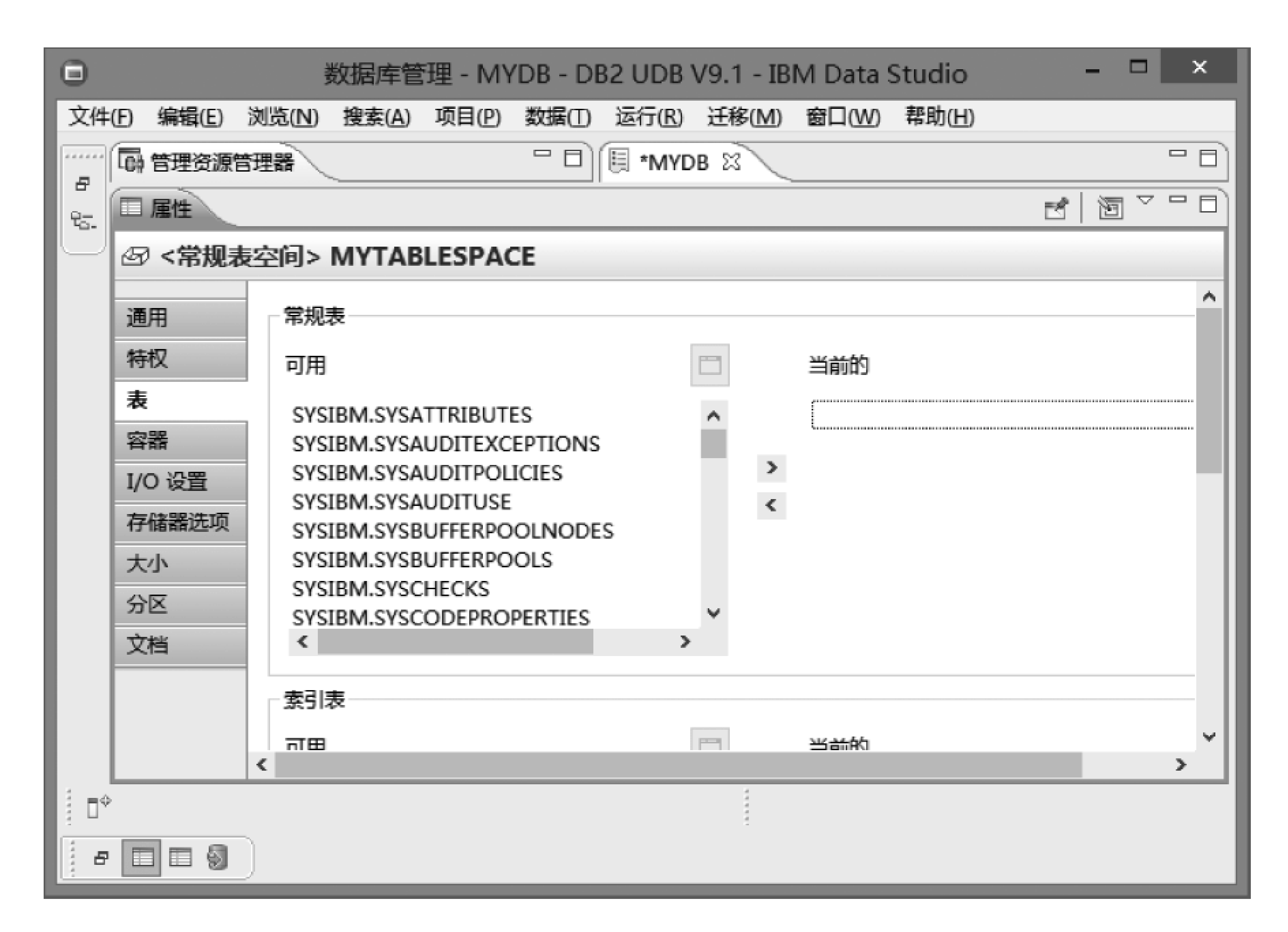


图 2.20 表空间的【表】设置



图 2.21 表空间的【容器】设置

(6) 对表空间的【I/O 设置】进行设置,如图 2.22 所示。



图 2.22　表空间的【I/O 设置】设置

(7) 对表空间的【存储器选项】进行设置,如图 2.23 所示。

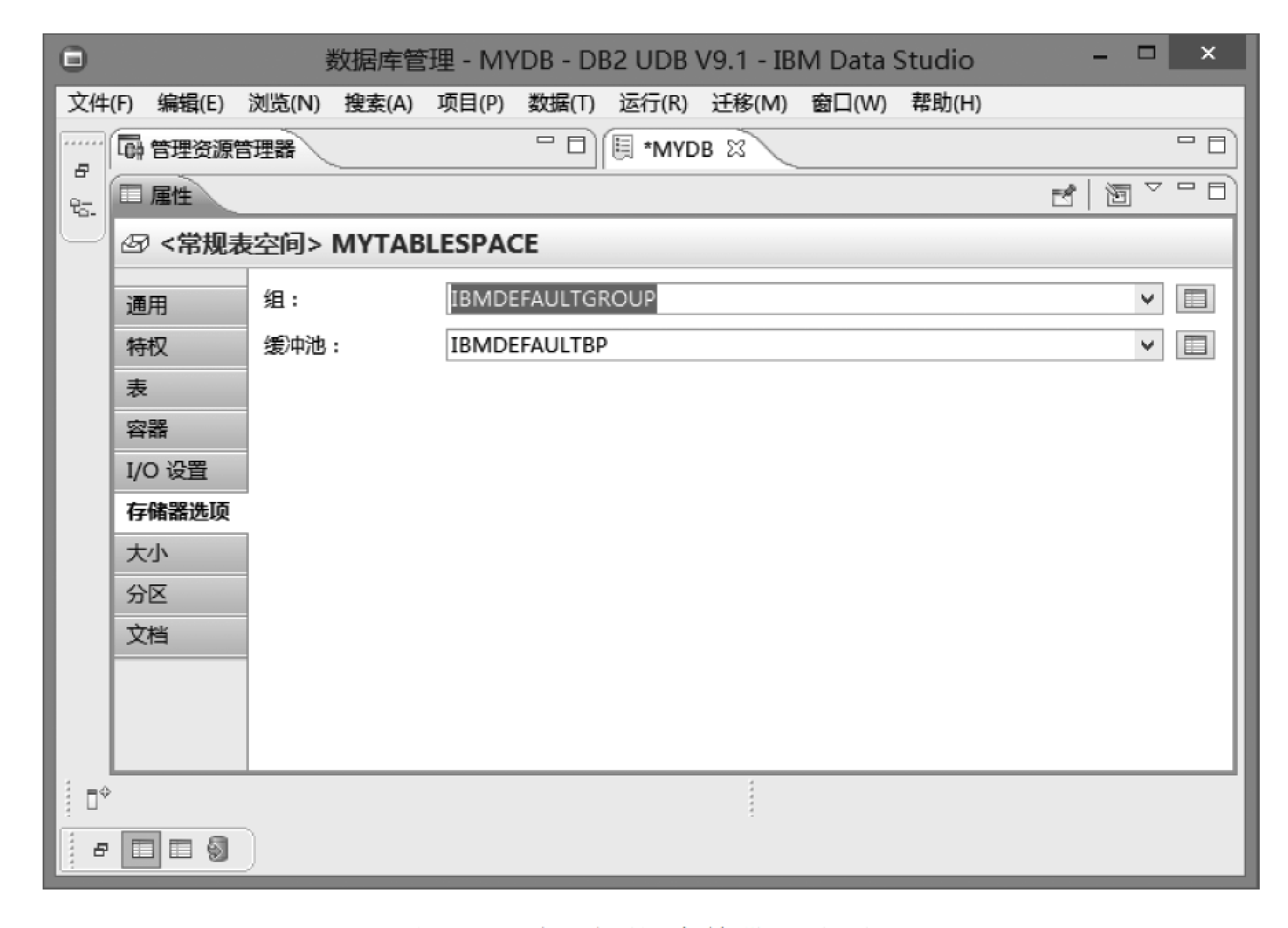


图 2.23　表空间的【存储器选项】设置

（8）对表空间的【大小】进行设置，如图 2.24 所示。



图 2.24　表空间的【大小】设置

（9）对表空间的【分区】进行设置，如图 2.25 所示。

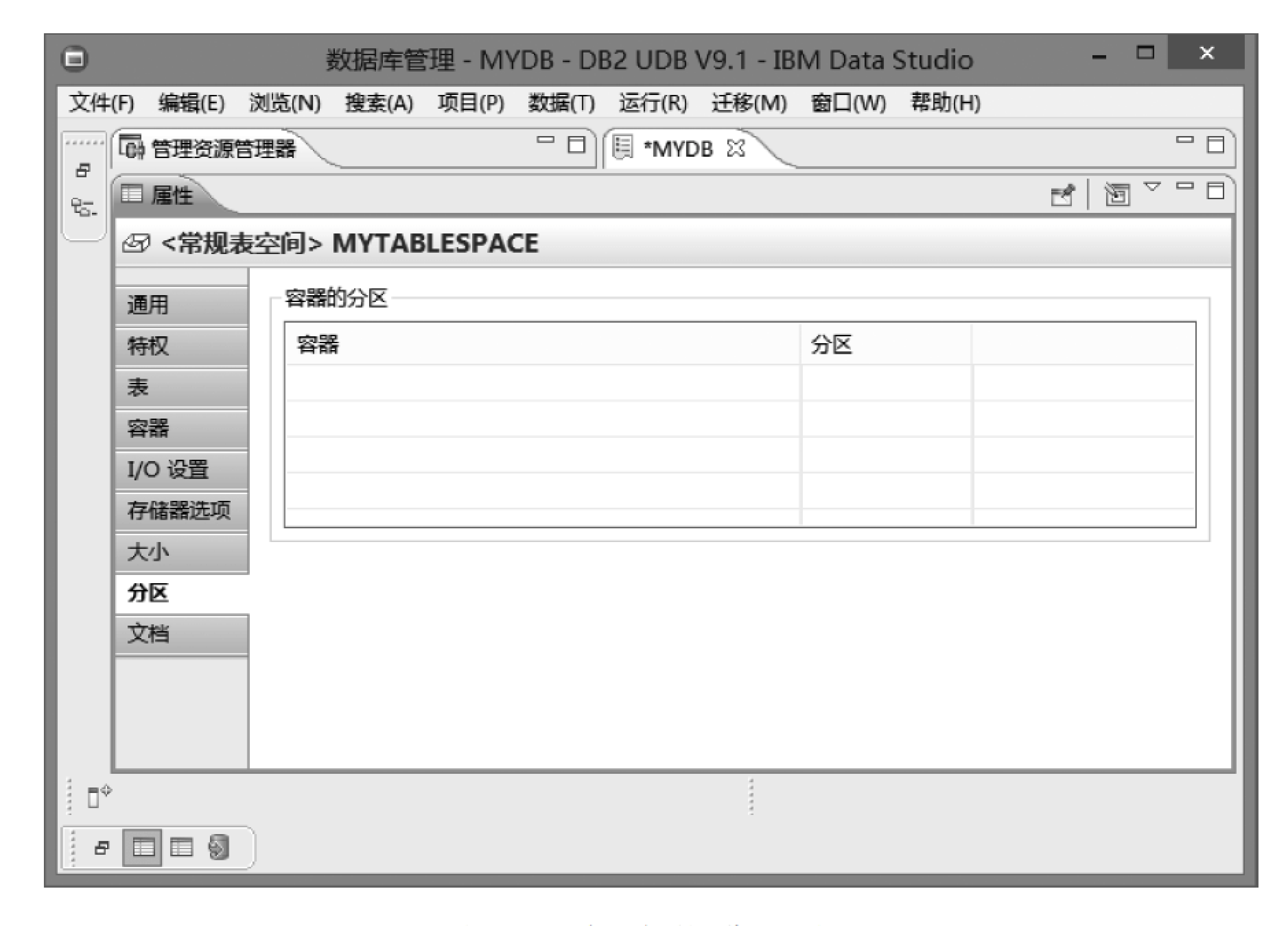


图 2.25　表空间的【分区】设置

(10) 对表空间的【文档】进行填写,如图 2.26 所示。



图 2.26 填写表空间的【文档】

(11) 对表空间进行部署,如图 2.27 所示。

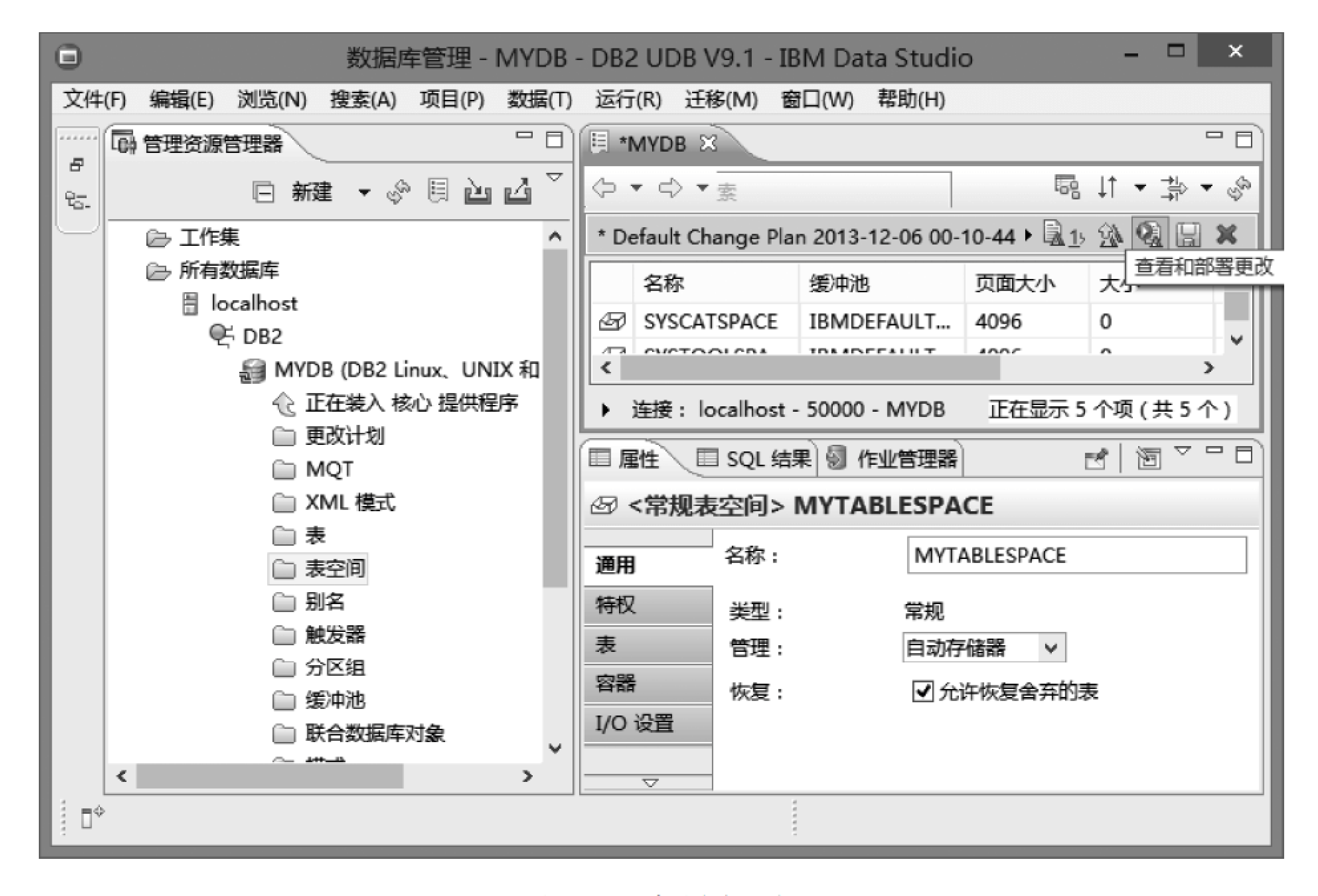


图 2.27 部署表空间

（12）单击【运行】按钮完成表空间创建，如图 2.28 所示。

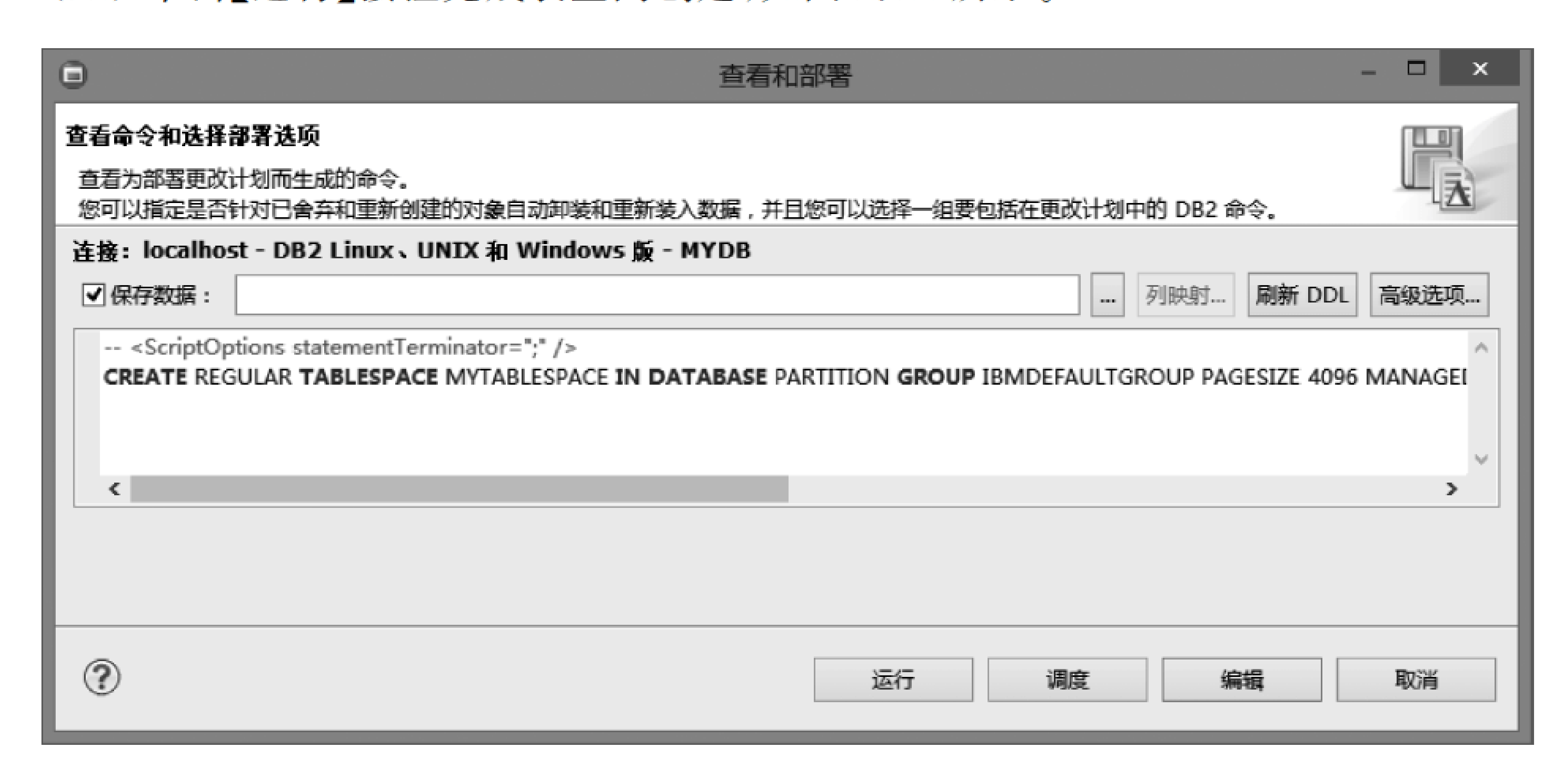


图 2.28 完成表空间创建

2.2.3 创建缓冲池

（1）右击选择【所有数据库】|localhost|DB2|MYDB|【缓冲池】|【创建缓冲池】命令，如图 2.29 所示。

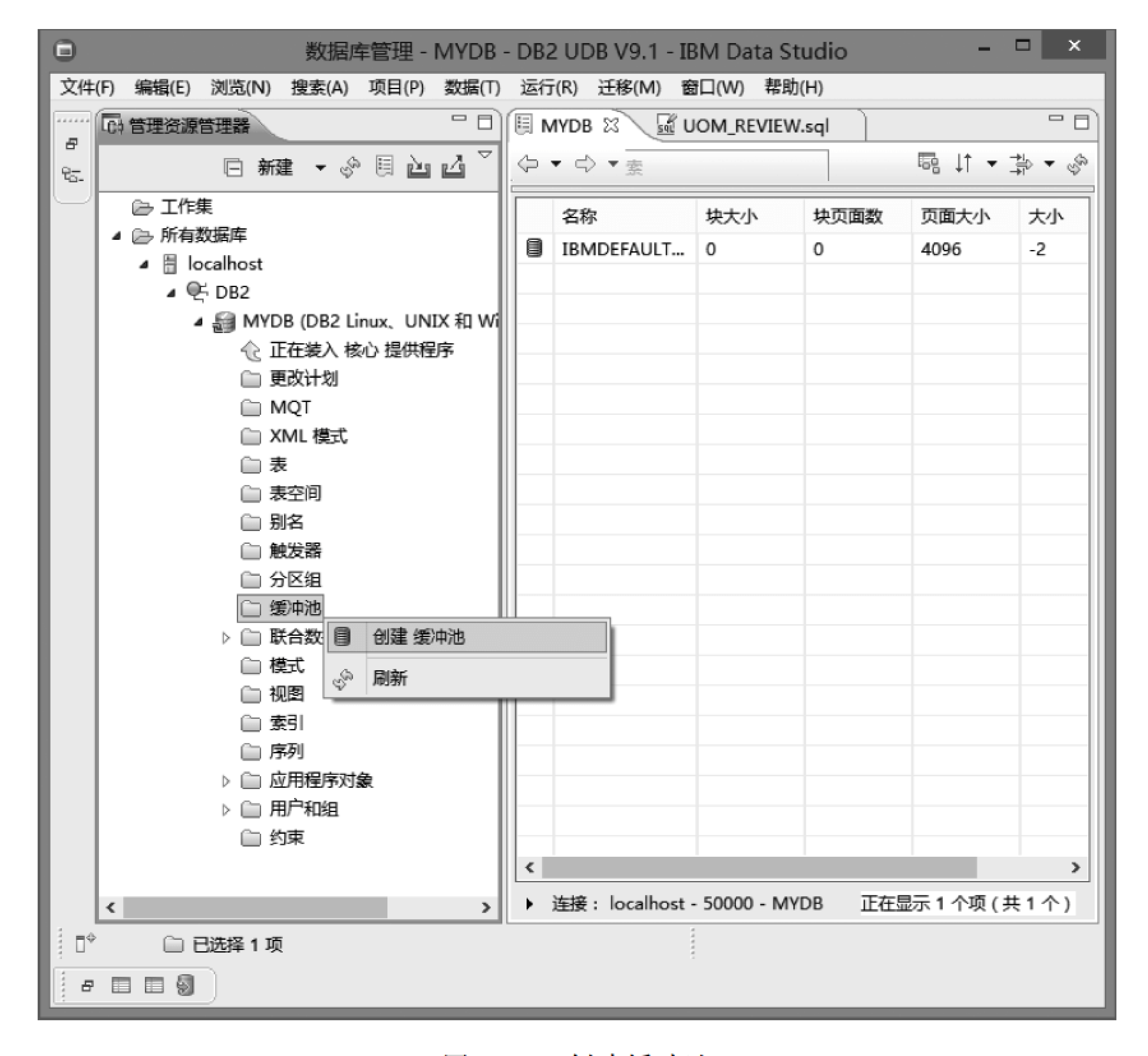


图 2.29 创建缓冲池

(2) 在【通用】选项卡中,对缓冲池“名称”等进行设置,如图 2.30 所示。

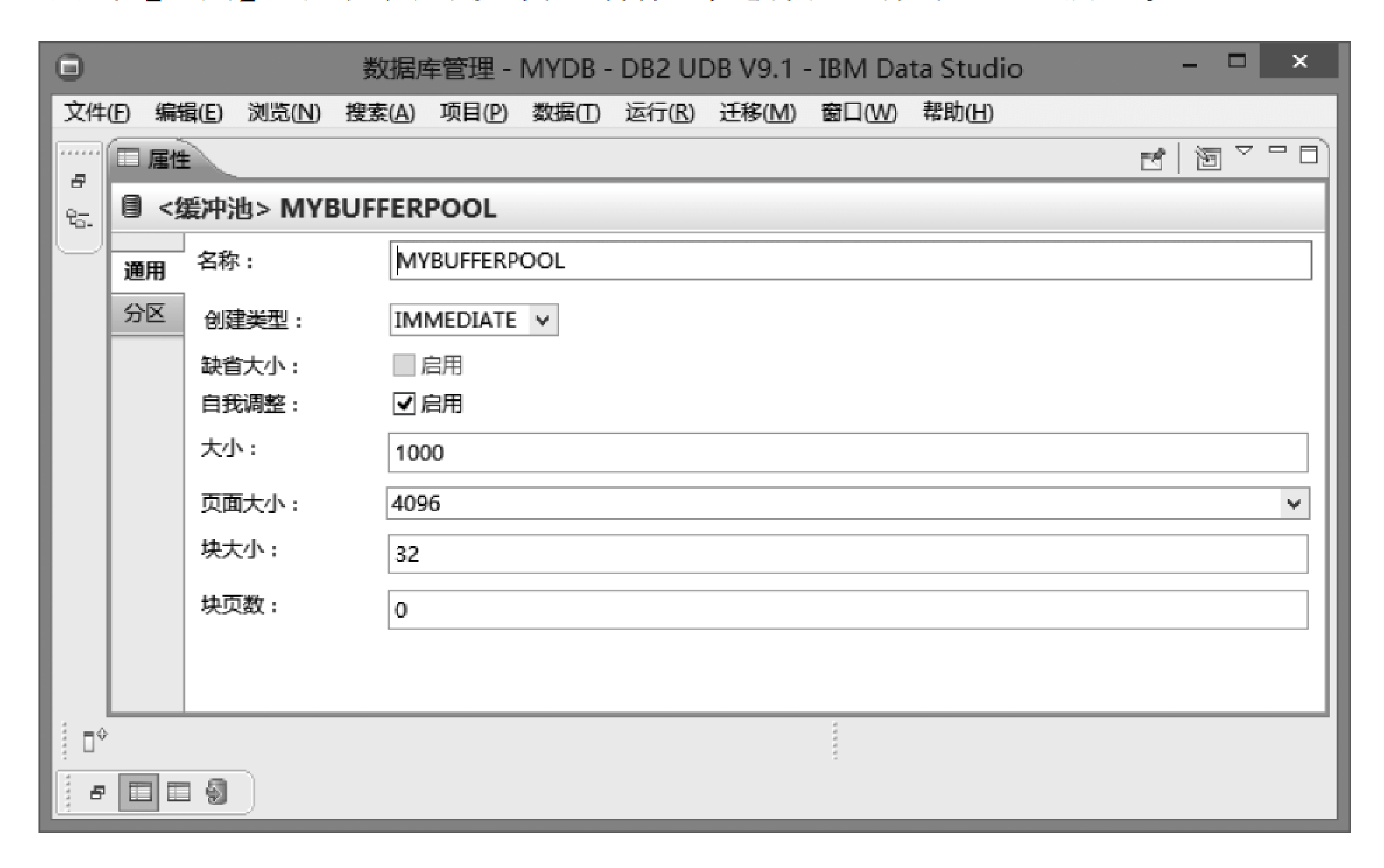


图 2.30 缓冲池的【通用】设置

(3) 对缓冲池【分区】进行设置,如图 2.31 所示。



图 2.31 缓冲池的【分区】设置

（4）单击【运行】按钮，完成缓冲池创建，如图 2.32 所示。

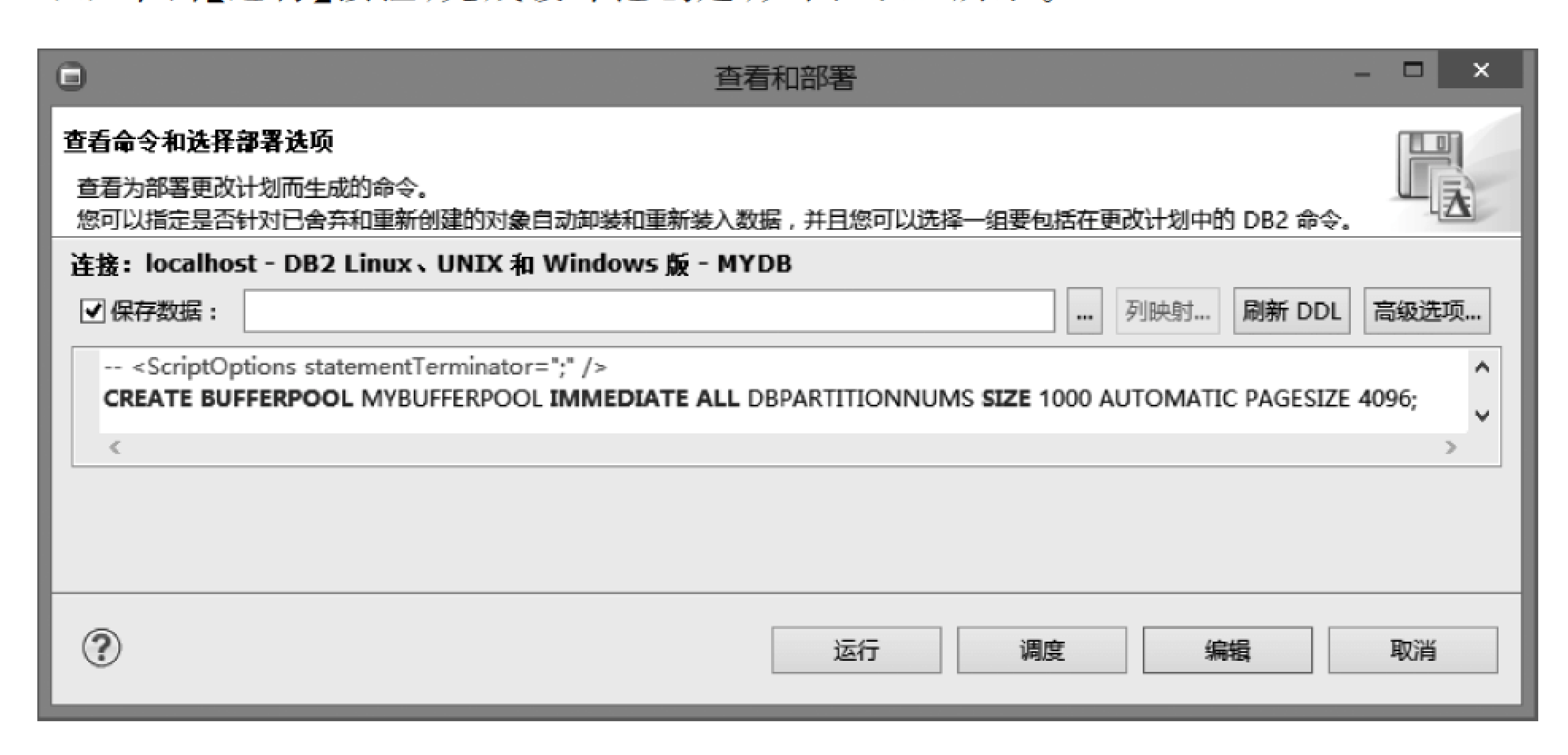


图 2.32 完成缓冲池创建

2.2.4 创建表

（1）右击选择【所有数据库】|localhost|DB2|MYDB|【表】|【创建表】命令，如图 2.33 所示。

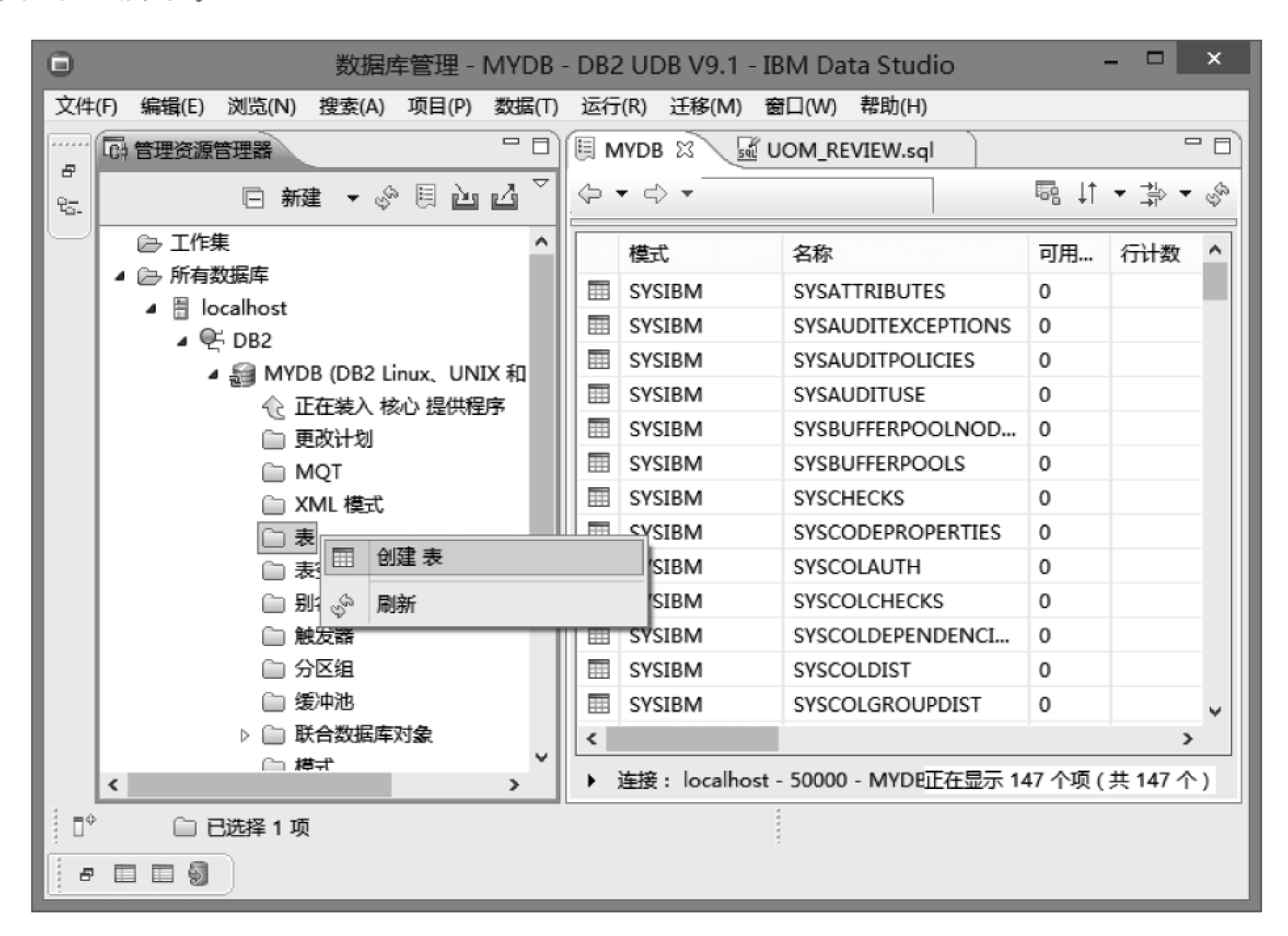


图 2.33 创建表

（2）在【选择模式】对话框中选择 MYSCHEMA 模式，单击【确定】按钮，如图 2.34 所示。

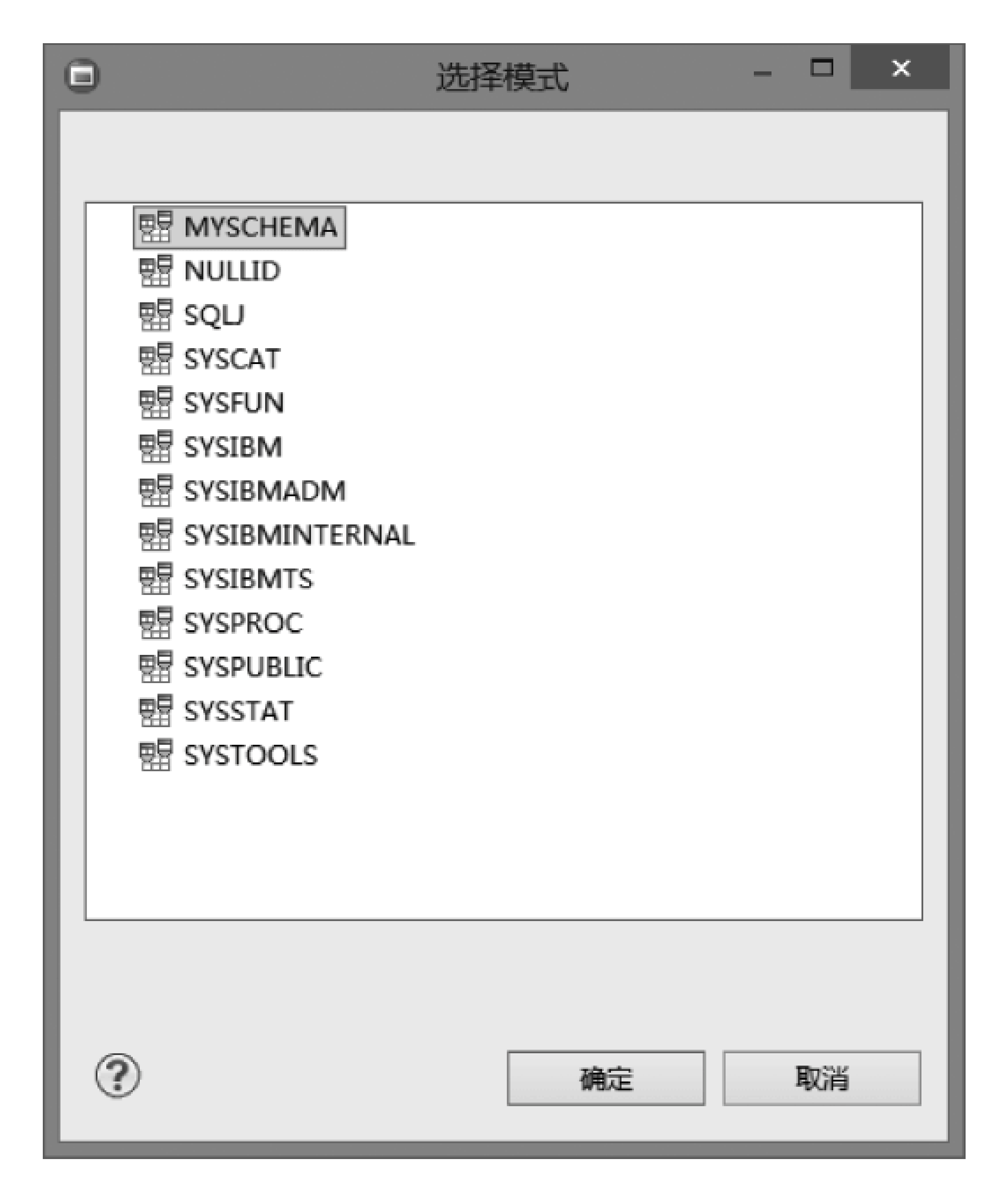


图 2.34　选择模式

(3) 进行表的【通用】设置，设置“名称”等信息，如图 2.35 所示。



图 2.35　表的【通用】设置

(4) 进行表的【列】设置，如图 2.36 所示。

(5) 进行表的【特权】设置，如图 2.37 所示。

(6) 进行表的【分布键】设置，如图 2.38 所示。

(7) 进行表的【数据分区】设置，如图 2.39 所示。



图 2.36　表的【列】设置

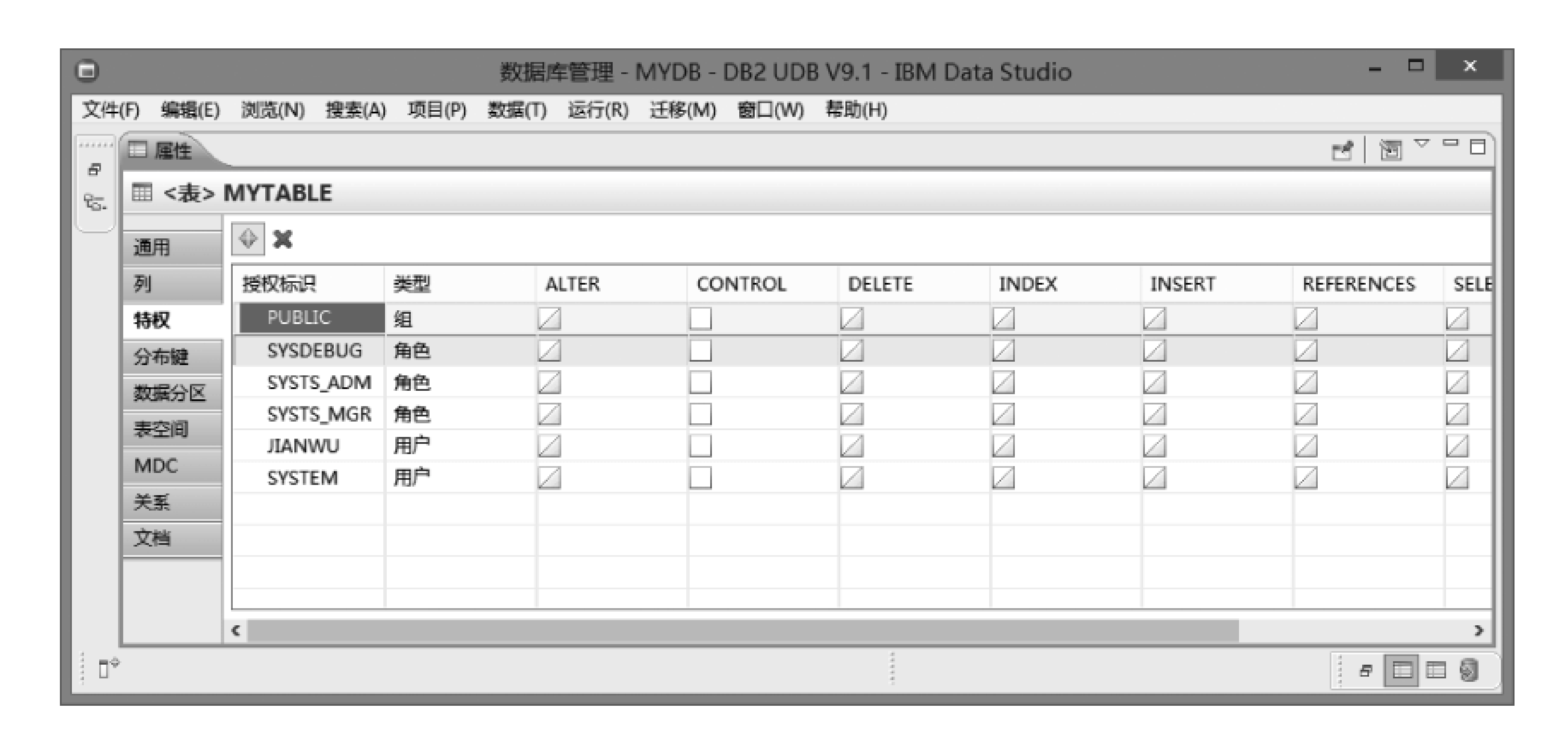


图 2.37　表的【特权】设置



图 2.38　表的【分布键】设置



图 2.39　表的【数据分区】设置

(8) 进行【选择列】操作，选择列后单击【确定】按钮，如图 2.40 所示。



图 2.40　选择列

(9) 进行表的【表空间】设置，如图 2.41 所示。

(10) 进行表的【MDC】设置，如图 2.42 所示。

(11) 进行表的【关系】设置，如图 2.43 所示。

(12) 对表的【文档】进行填写，如图 2.44 所示。



图 2.41　表的【表空间】设置



图 2.42　表的【MDC】设置



图 2.43 表的【关系】设置



图 2.44 表的【文档】填写

（13）对表进行部署，如图 2.45 所示。

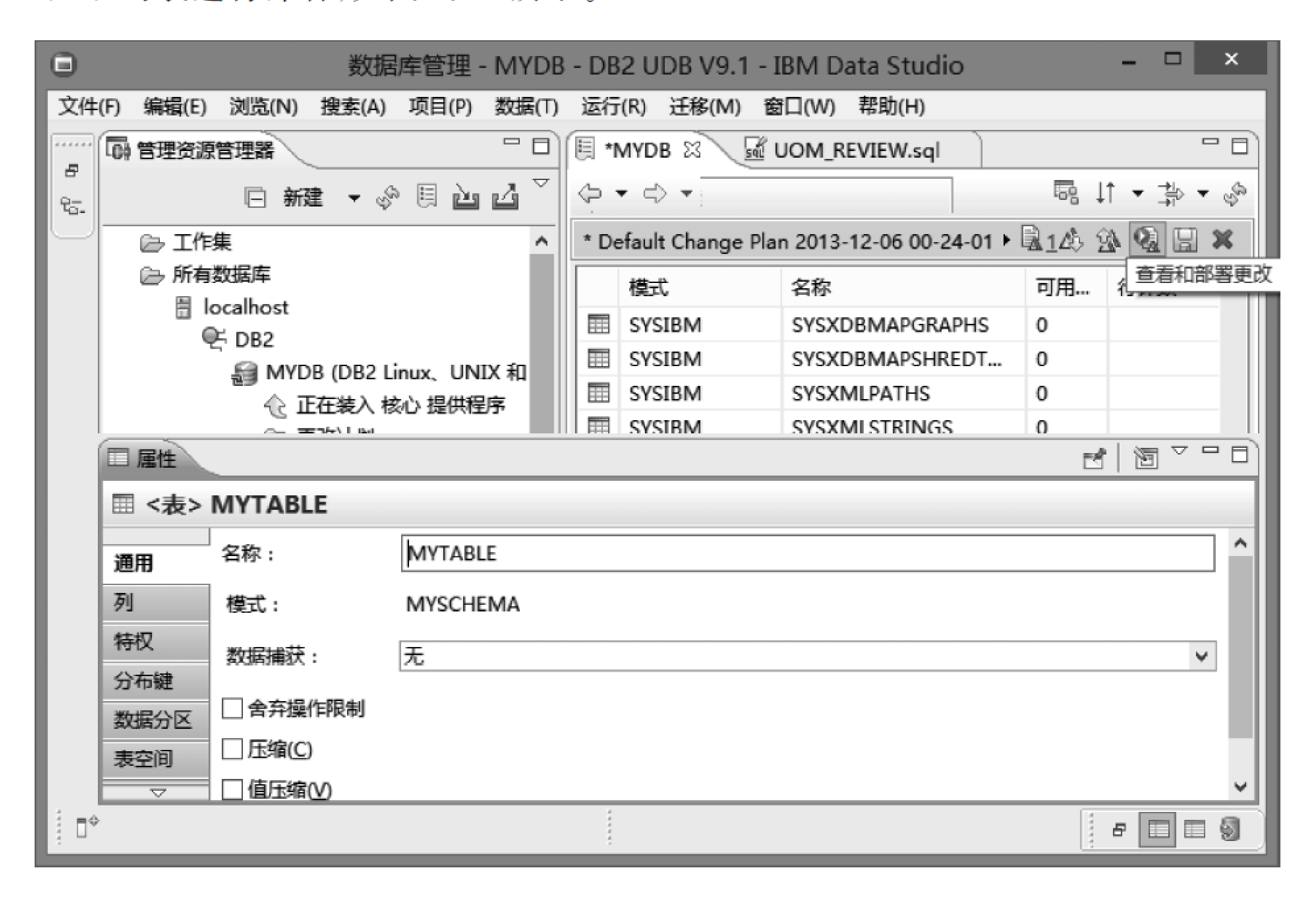


图 2.45　对表进行部署

（14）单击【运行】按钮，完成表的创建，如图 2.46 所示。



图 2.46　完成表的部署

2.2.5　创建索引

（1）右击选择【所有数据库】|localhost|DB2|MYDB|【索引】|【创建索引】命令，如图 2.47 所示。

（2）进行【选择表】操作，如图 2.48 所示。

（3）对索引进行【通用】设置，如图 2.49 所示。

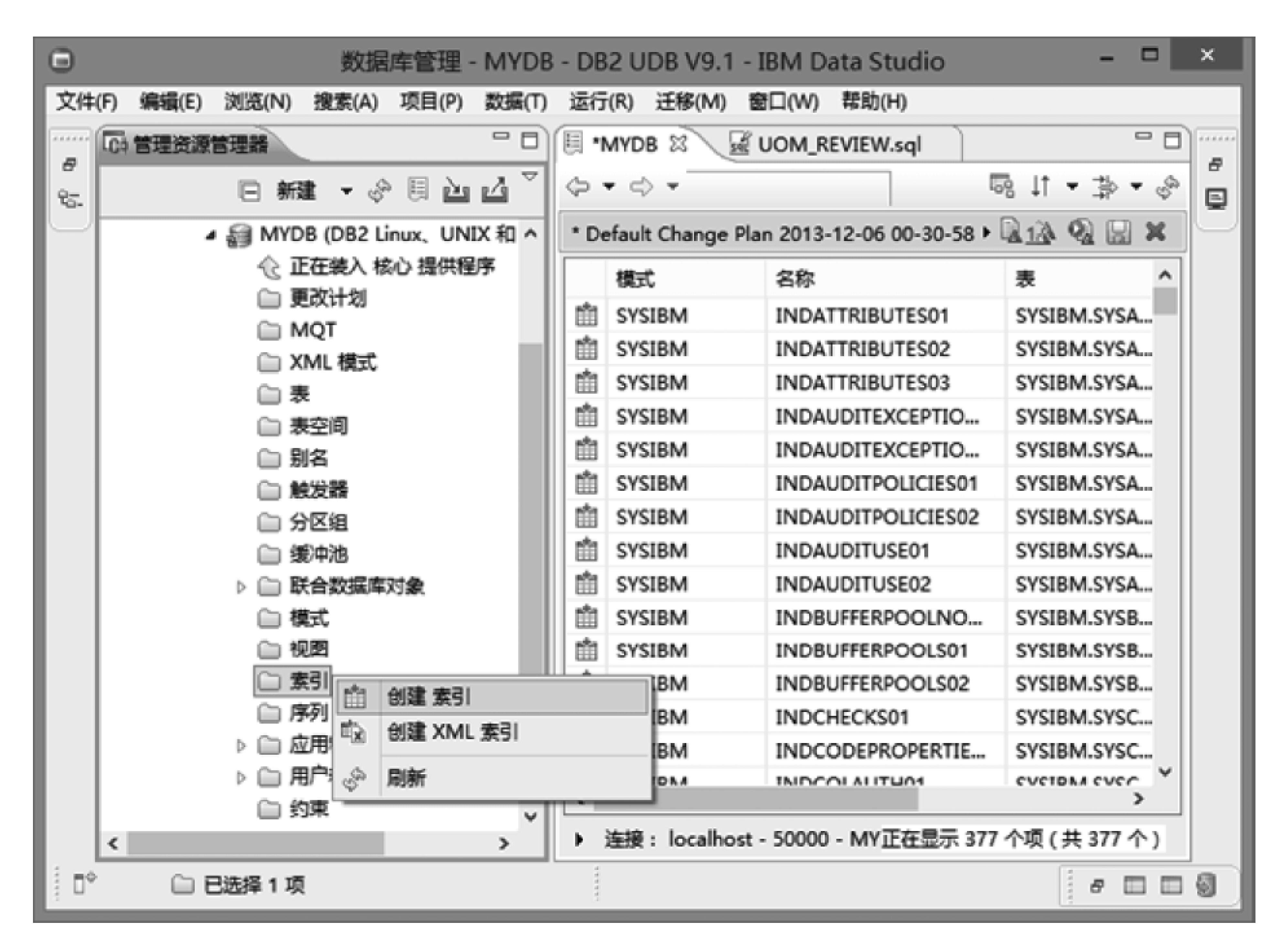


图 2.47 创建索引

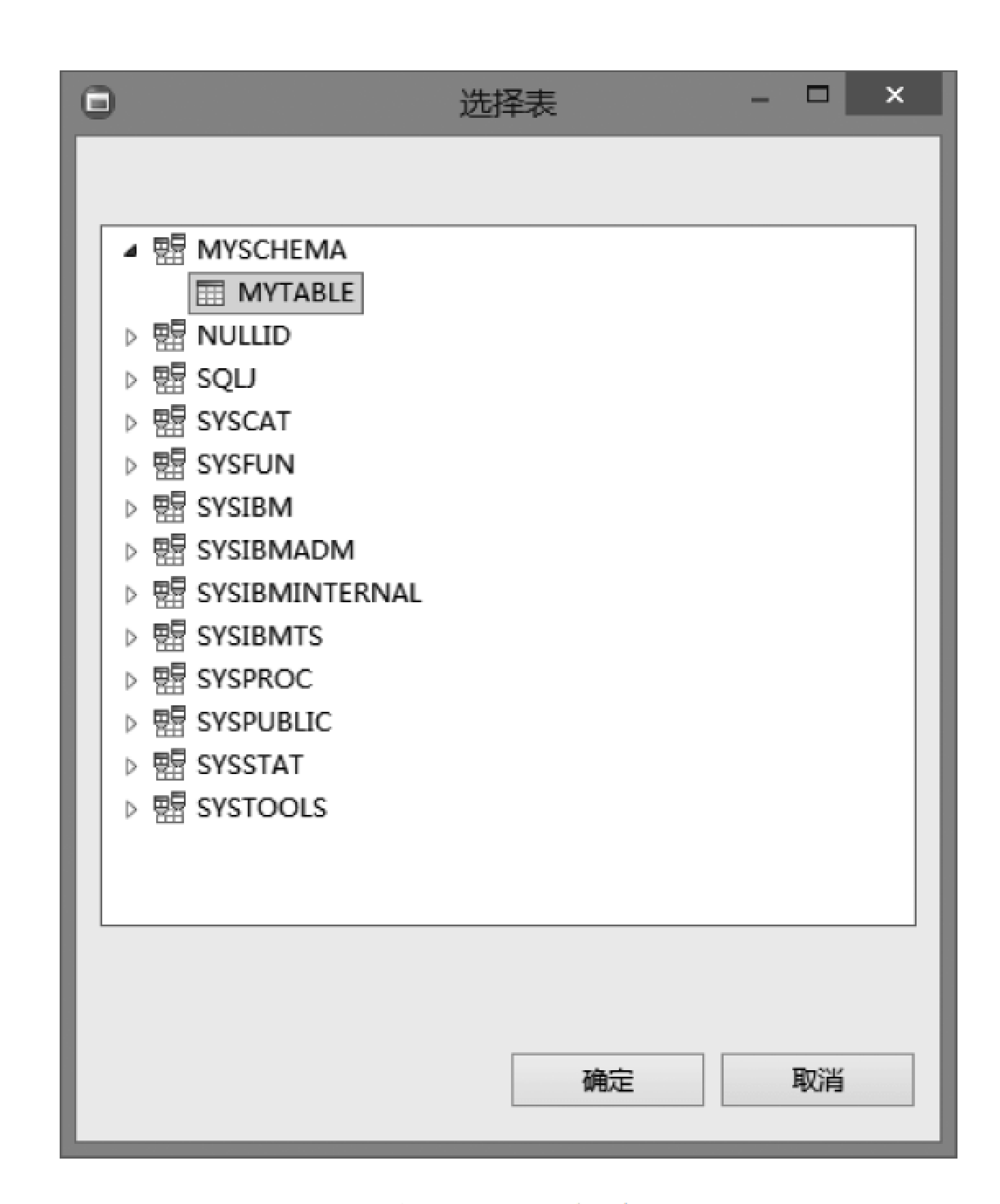


图 2.48 选择表

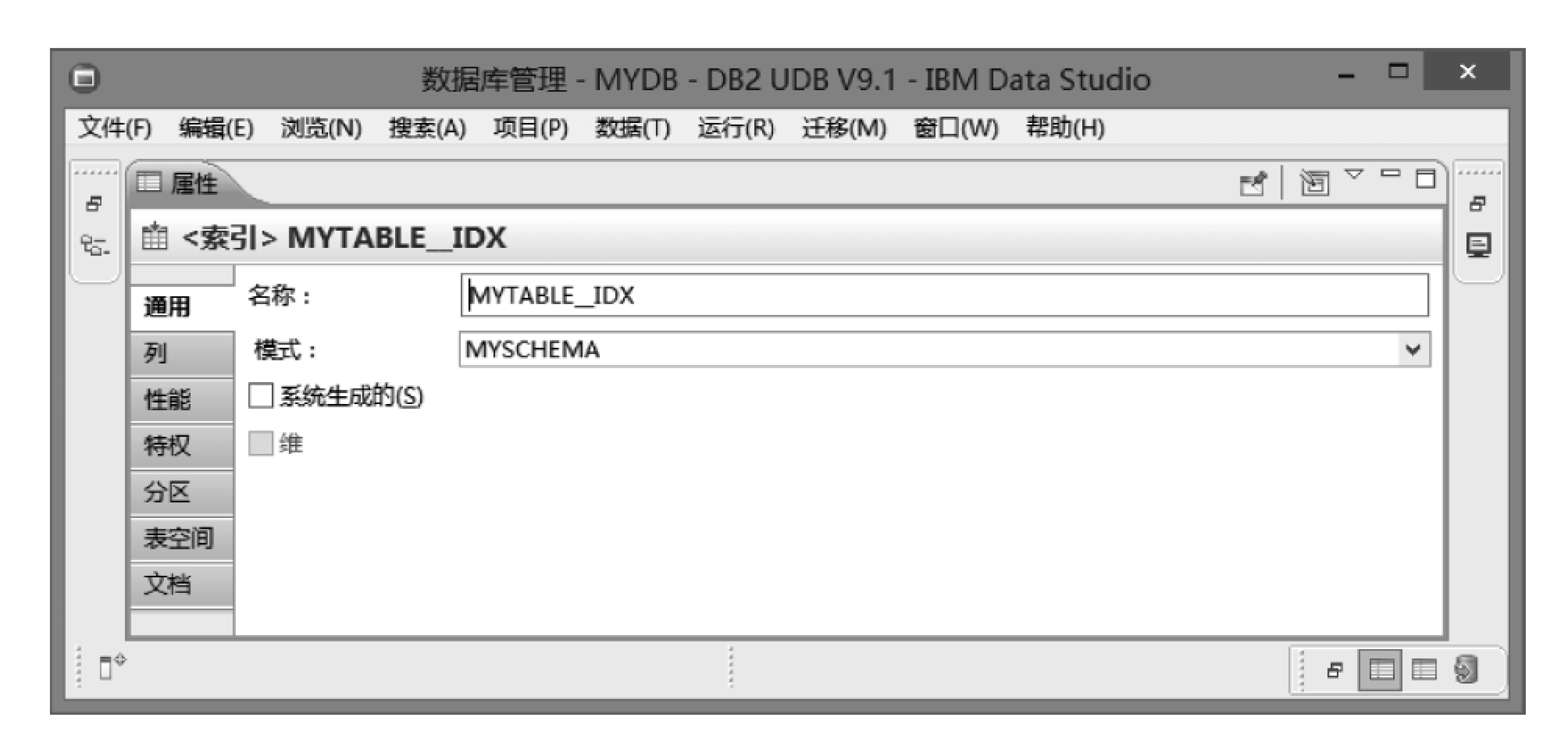


图 2.49 索引【通用】设置

(4) 对索引进行【列】设置，如图 2.50 所示。



图 2.50 索引【列】设置

(5) 进行【选择列】操作，单击【确定】按钮，如图 2.51 所示。

(6) 进行索引【性能】设置，如图 2.52 所示。

(7) 进行索引【特权】设置，如图 2.53 所示。

(8) 进行索引【分区】设置，如图 2.54 所示。

(9) 进行索引【表空间】设置，如图 2.55 所示。

(10) 进行索引【文档】填写，如图 2.56 所示。

(11) 进行索引部署，如图 2.57 所示。

(12) 单击【运行】按钮完成索引的创建，如图 2.58 所示。



图 2.51　选择列



图 2.52　索引【性能】设置

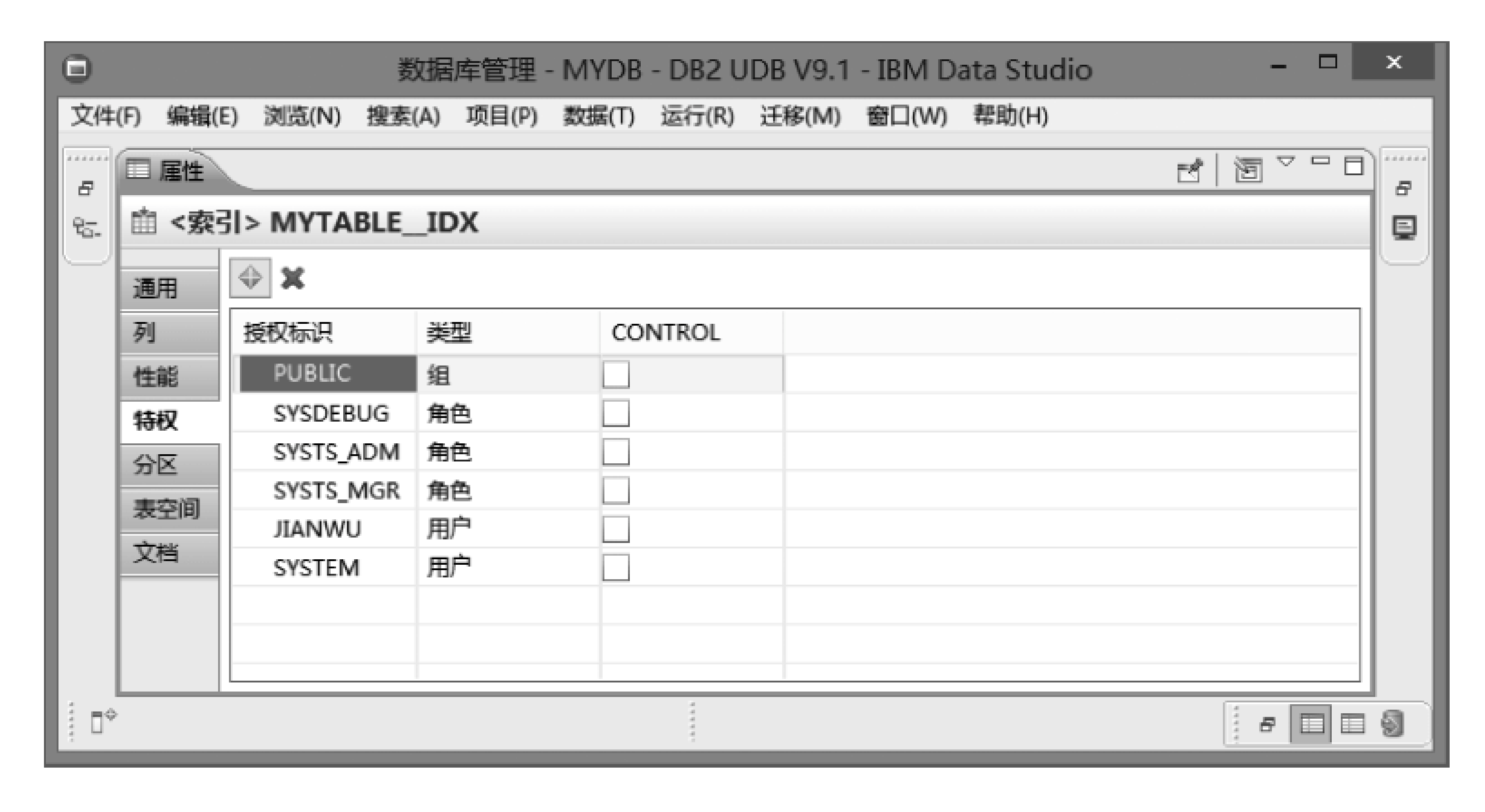


图 2.53　索引【特权】设置



图 2.54　索引【分区】设置



图 2.55　索引【表空间】设置



图 2.56 索引【文档】填写



图 2.57 部署索引



图 2.58 完成索引的创建

2.2.6　创建视图

（1）右击选择【所有数据库】|localhost|DB2|MYDB|【视图】|【创建视图】命令，如图 2.59 所示。

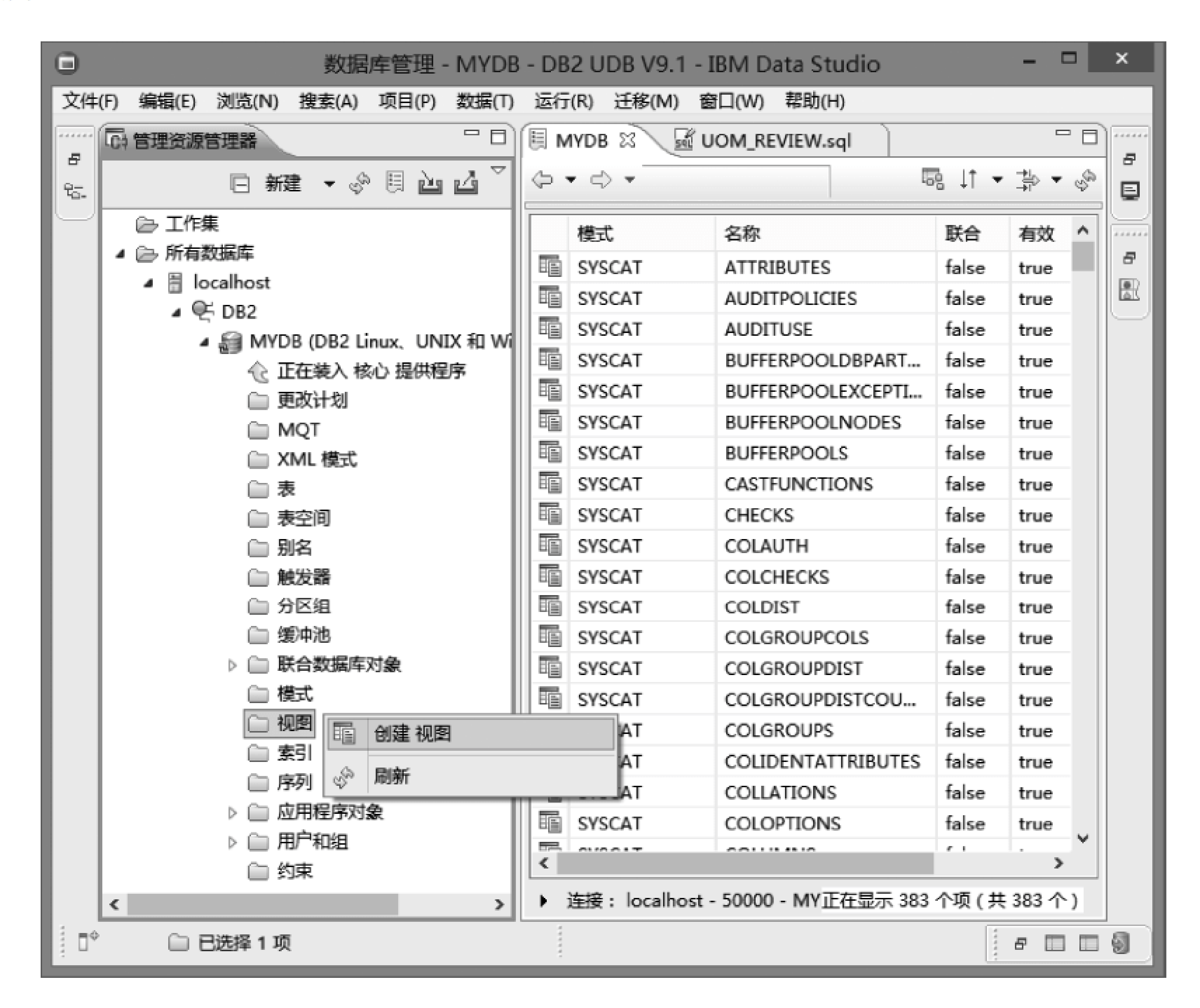


图 2.59　创建视图

（2）进行选择模式，如图 2.60 所示。

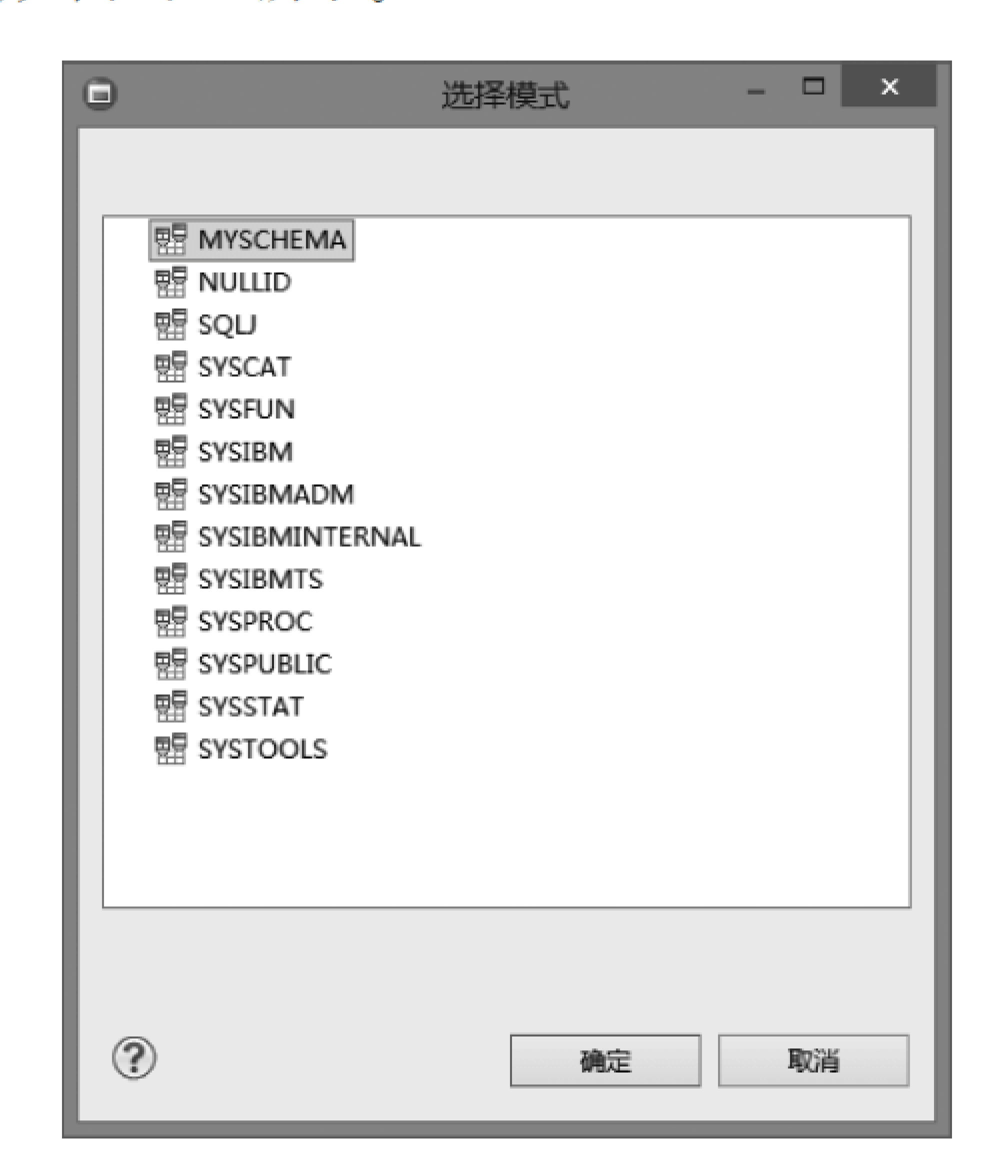


图 2.60　选择模式

(3) 进行视图【通用】设置，如图 2.61 所示。



图 2.61 视图【通用】设置

(4) 进行视图【列】设置，如图 2.62 所示。



图 2.62 视图【列】设置

(5) 进行视图【特权】设置，如图 2.63 所示。

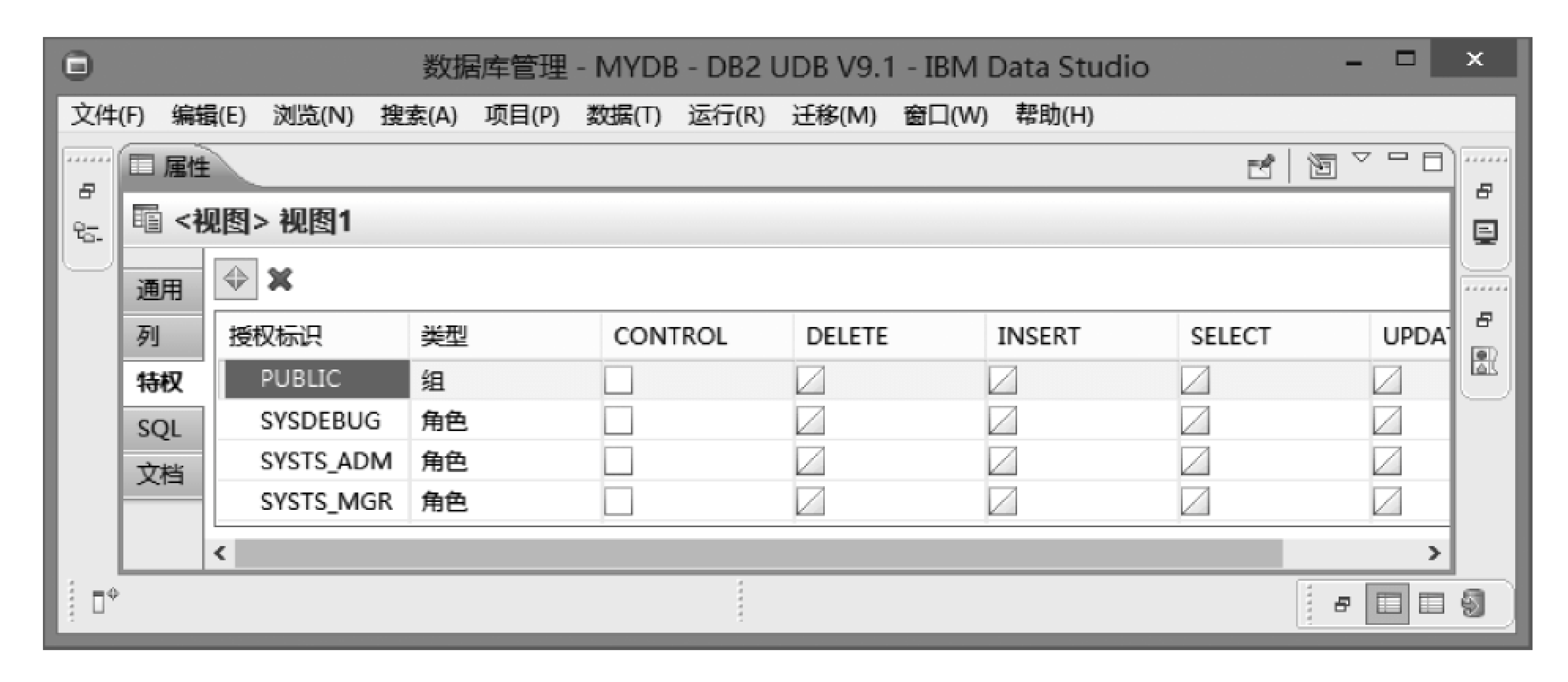


图 2.63 视图【特权】设置

(6) 进行视图 SQL 设置，如图 2.64 所示。

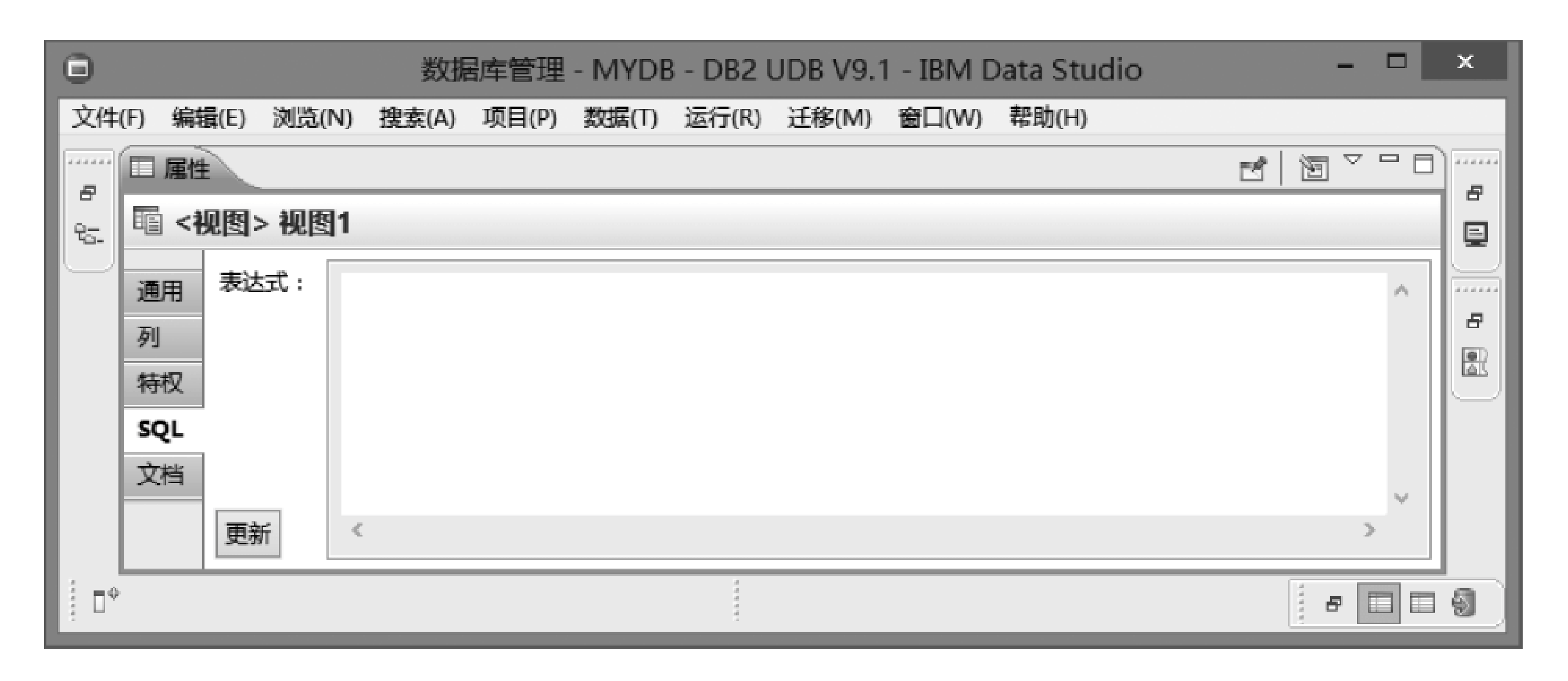


图 2.64　视图【SQL】设置

(7) 填写视图【文档】，如图 2.65 所示。



图 2.65　填写视图通用【文档】

(8) 进行视图部署操作，如图 2.66 所示。

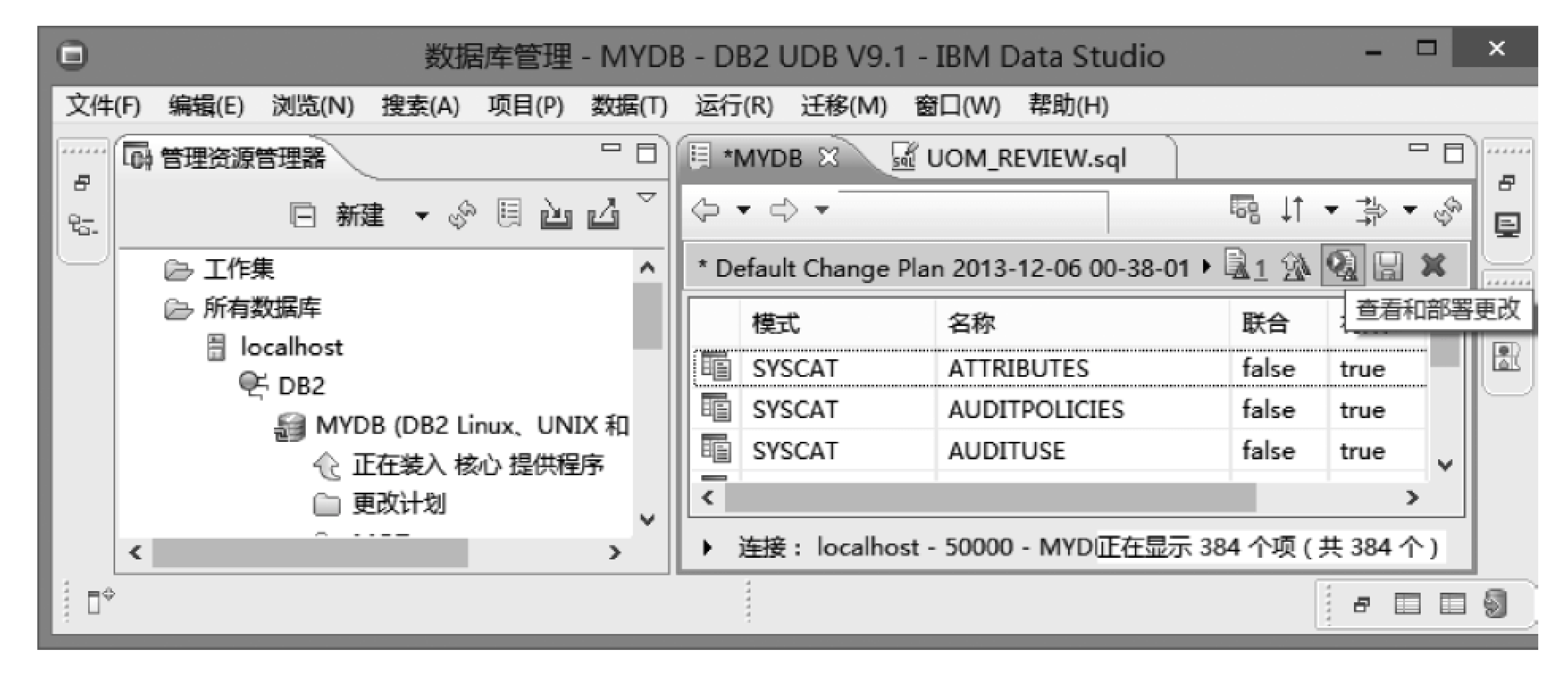


图 2.66　部署视图

2.2.7 创建别名

右击选择【所有数据库】| localhost | DB2 | MYDB |【别名】|【创建别名】命令，如图 2.67 所示。

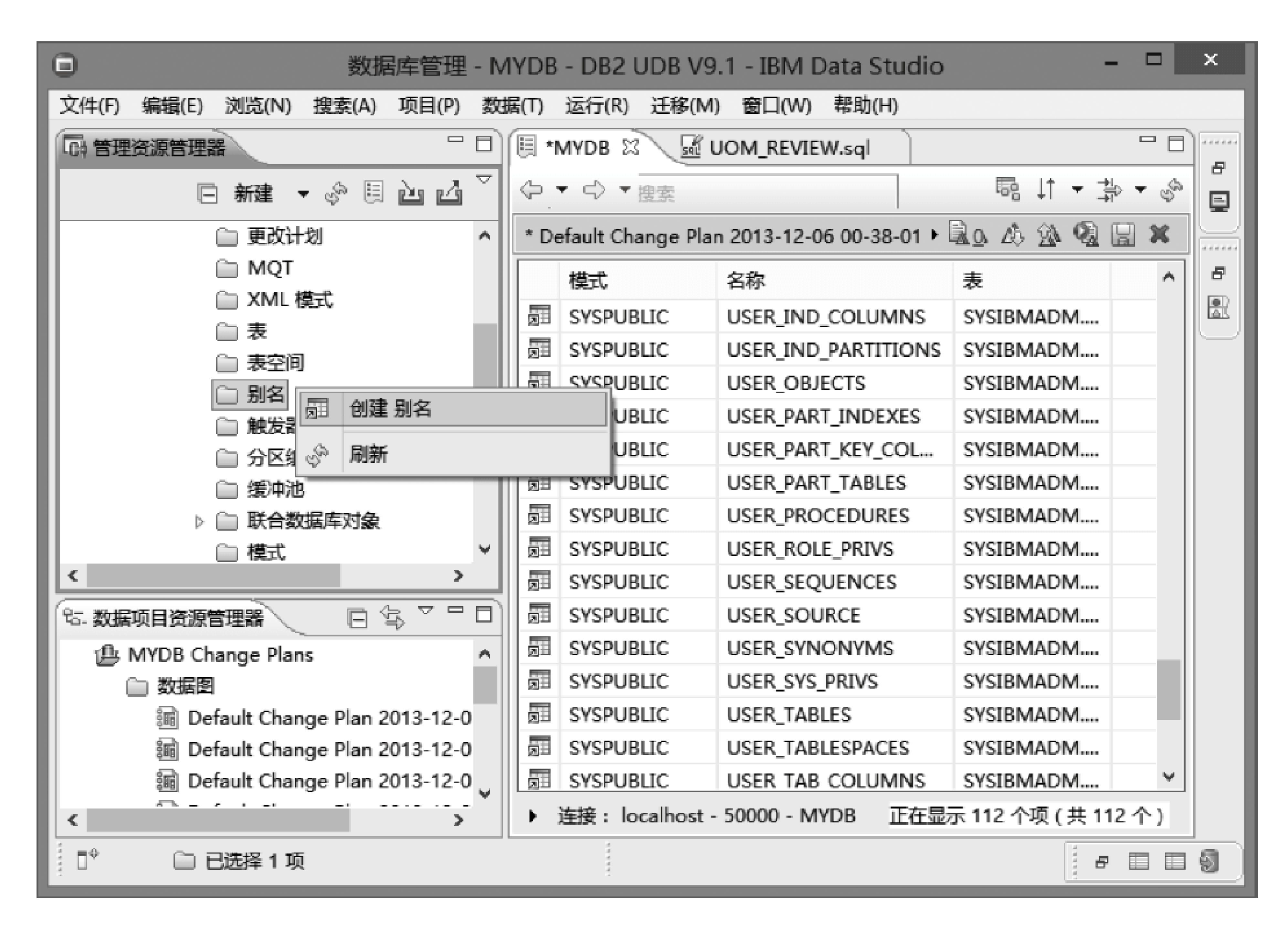


图 2.67 创建别名

其余步骤尝试自行操作。

2.2.8 创建约束

右击选择【所有数据库】| localhost | DB2 | MYDB |【约束】|【创建唯一约束】命令，如图 2.68 所示。

其余步骤尝试自行操作。

2.2.9 创建触发器

右击选择【所有数据库】| localhost | DB2 | MYDB |【触发器】|【创建触发器】命令，如图 2.69 所示。

其余步骤尝试自行操作。

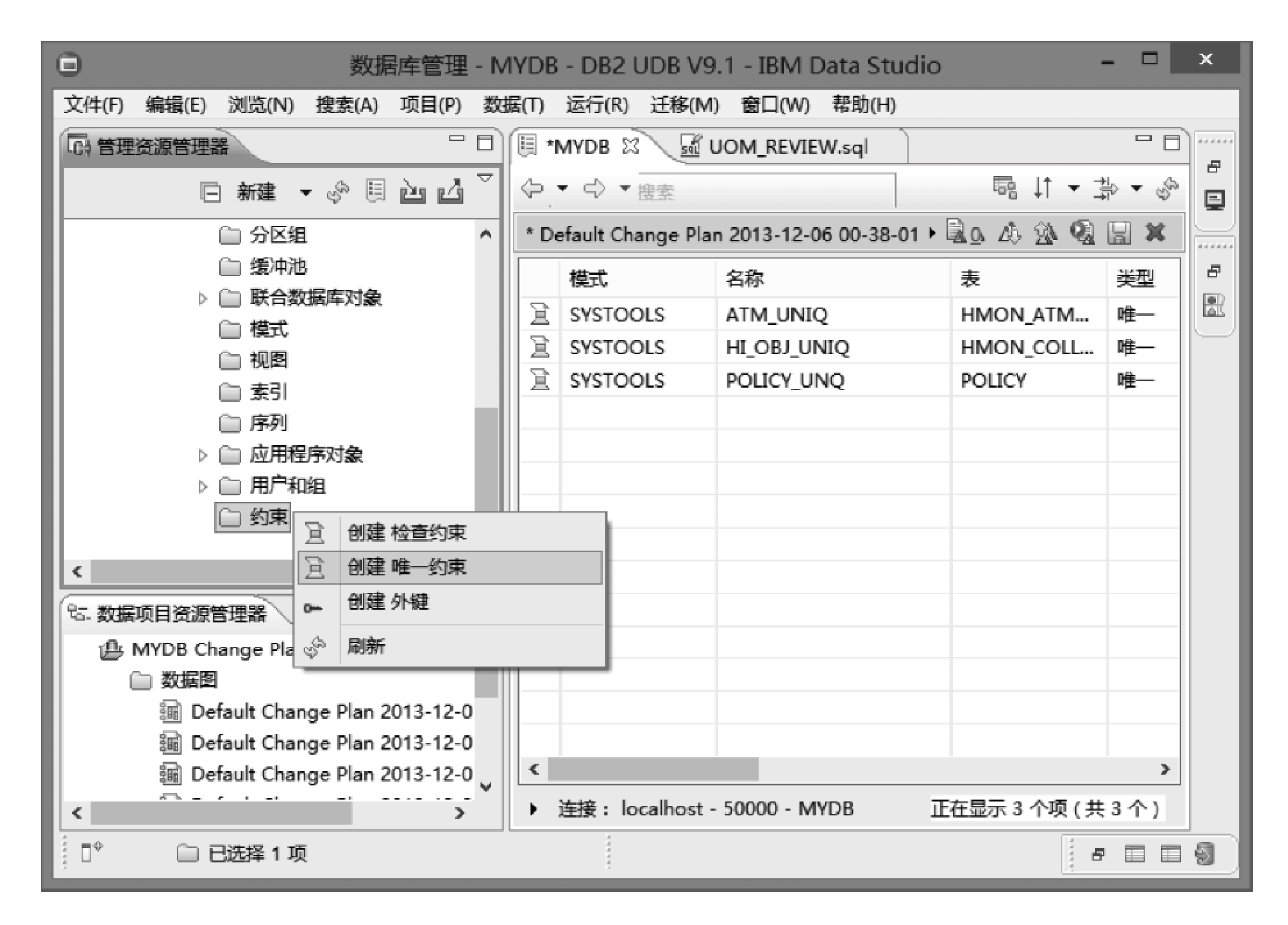


图 2.68　创建约束

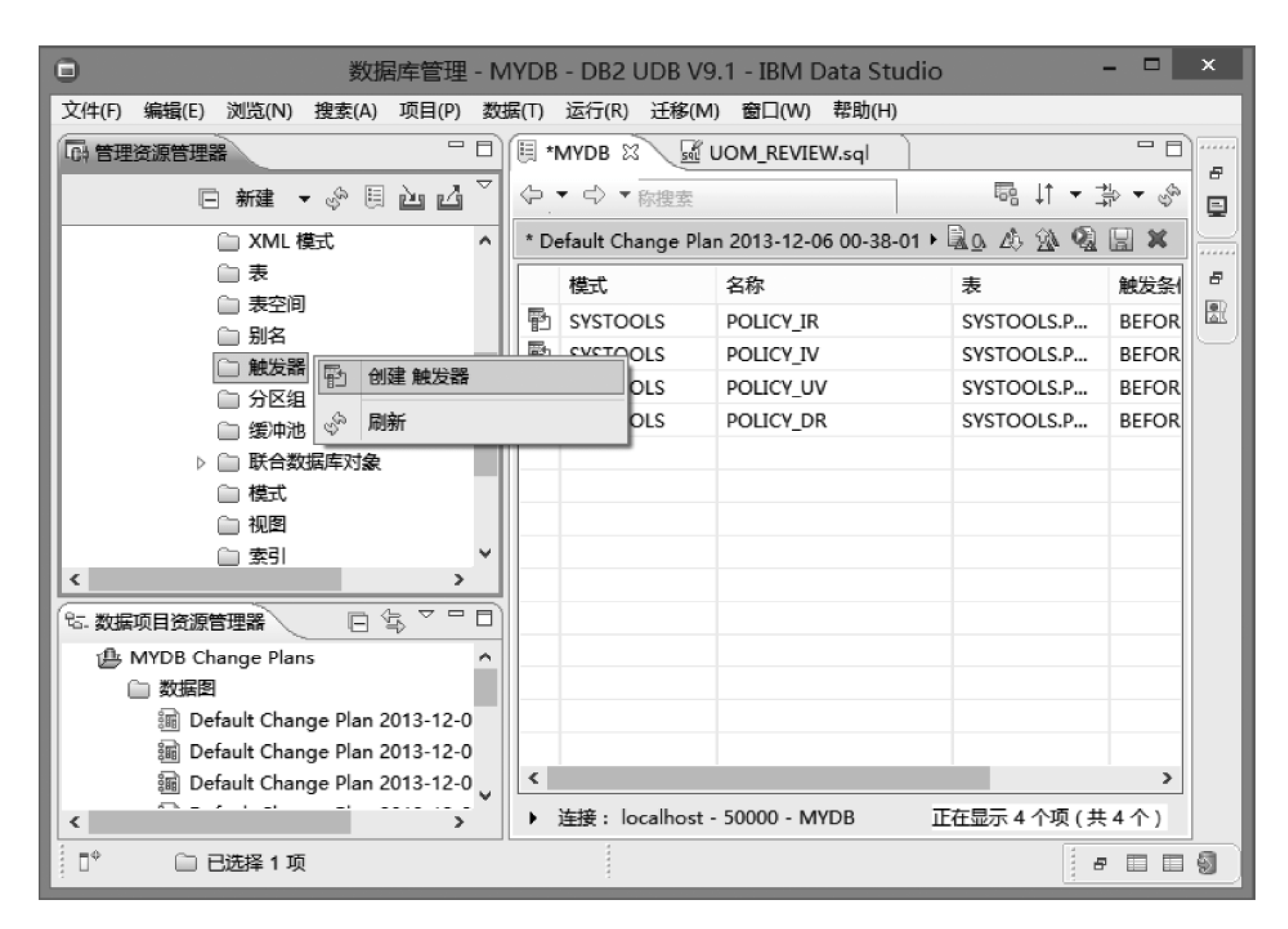


图 2.69　创建触发器

2.3 备份和恢复

2.3.1 DB2 数据库的备份

(1) 右击选择【所有数据库】|localhost|DB2|MYDB|【备份和复原】|【备份… 】命令，如图 2.70 所示。

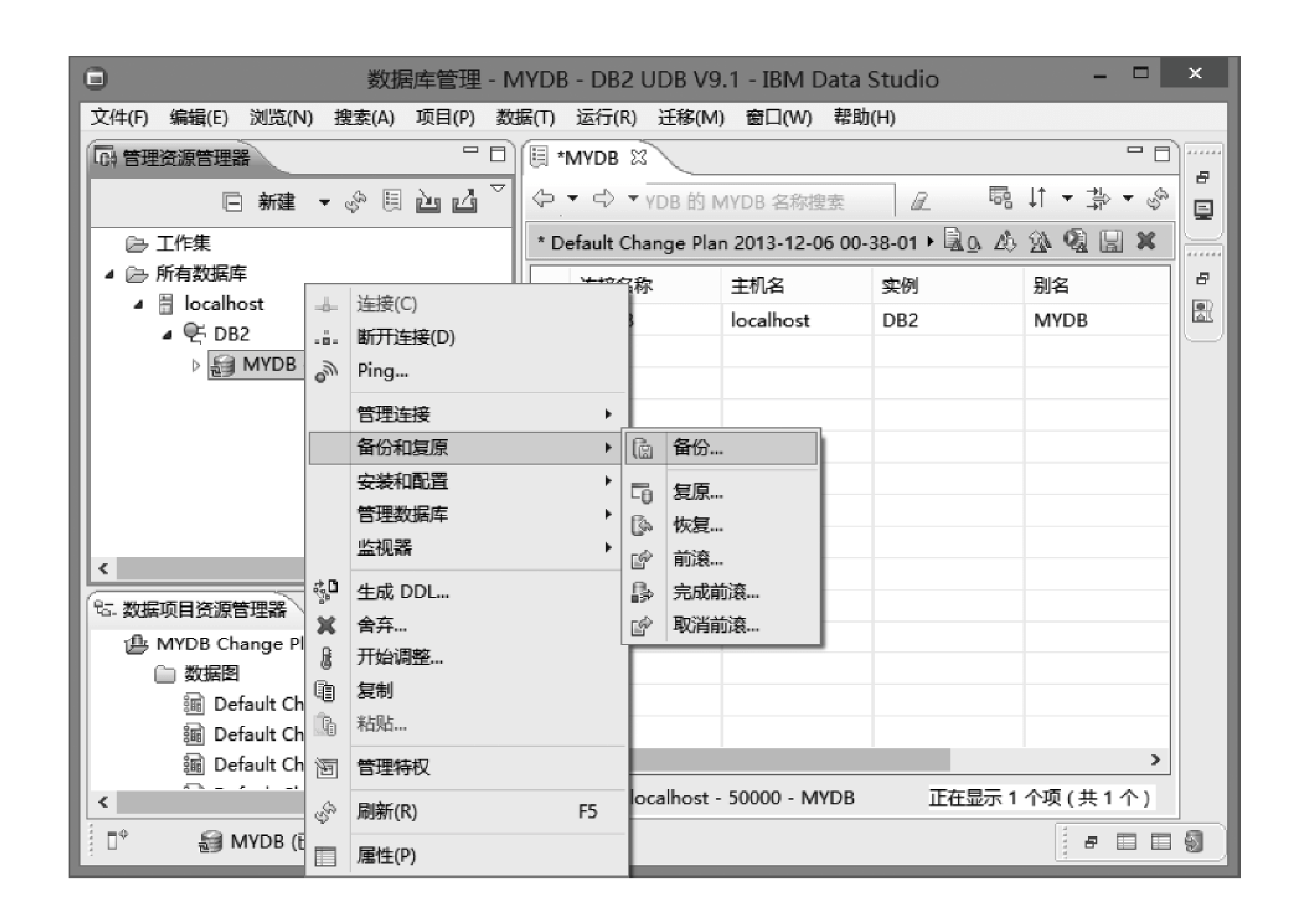


图 2.70 数据库备份

(2) 对【备份信息】进行确认，如图 2.71 所示。

(3) 为备份指定类型选项，如图 2.72 所示。

(4) 指定存储备份映像的位置，如图 2.73 所示。

(5) 选择位置后，单击【确定】按钮，如图 2.74 所示。

(6) 选择【备份选项】，如图 2.75 所示。

(7) 进行【备份性能】设置，如图 2.76 所示。

(8) 最后单击【运行】按钮完成备份。



图 2.71　备份信息



图 2.72　备份类型



图 2.73 备份映像位置



图 2.74 选择位置



图 2.75 备份选项



图 2.76 备份性能

2.3.2 DB2 数据库的恢复

（1）右击选择【所有数据库】|localhost|DB2|MYDB|【备份和复原】|【复原】命令，如图 2.77 所示。

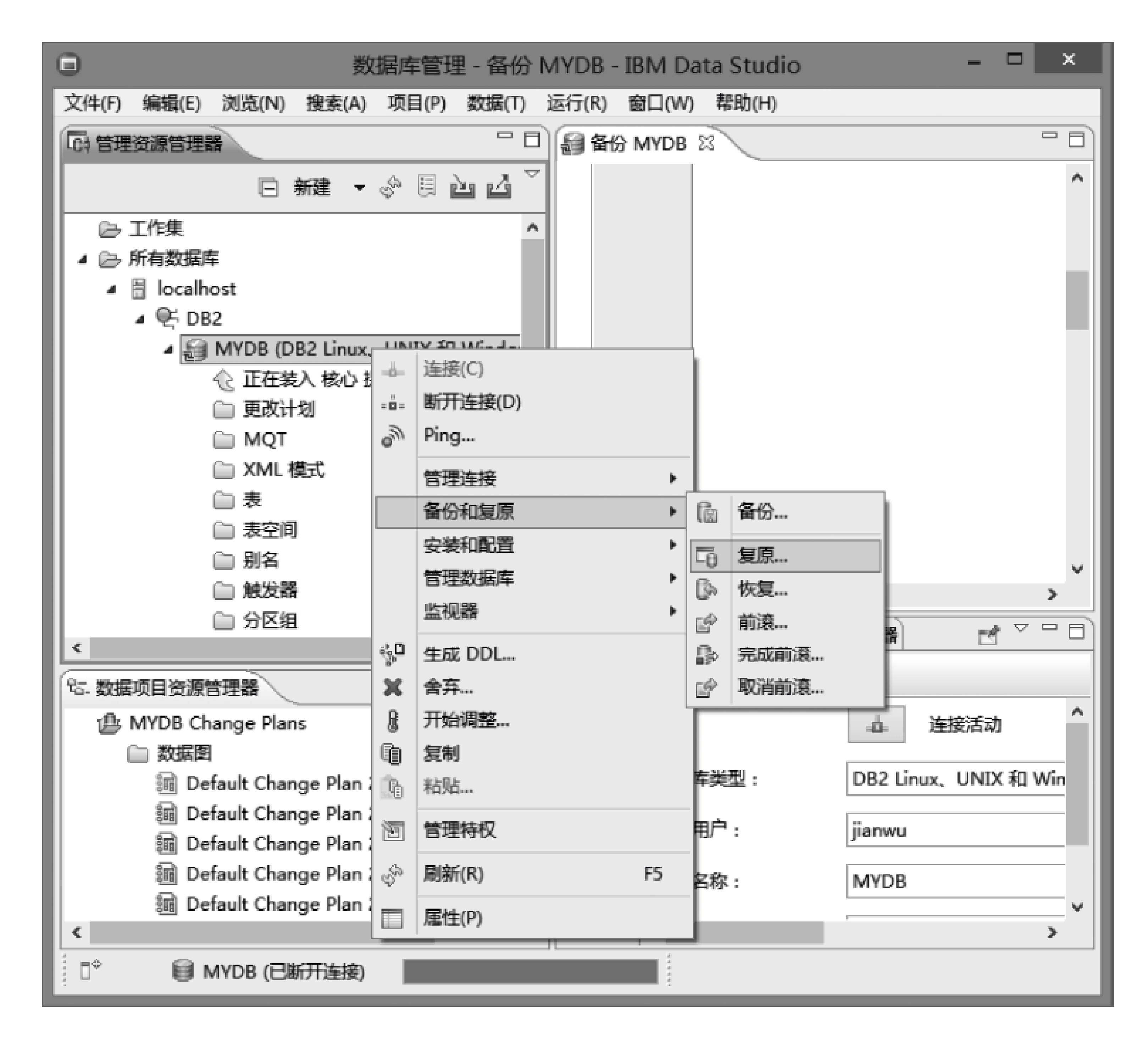


图 2.77 数据库复原

（2）选择【复原类型】，如图 2.78 所示。



图 2.78 复原类型

(3) 选择数据库对象和备份映像,如图 2.79 所示。



图 2.79 复原对象

(4) 对【复原容器】进行设置,如图 2.80 所示。



图 2.80 复原容器

（5）对【复原选项】进行设置，如图 2.81 所示。



图 2.81　备份复原选项

（6）单击【运行】按钮完成数据库复原。

2.4　DB2 优化器

2.4.1　优化器

这里以一个语句来开始 DB2 优化器的学习，这个语句并不是以读者熟悉的 SQL 如“select c1，c2 from…”的方式给出，而是以读者更加熟悉的陈述方式给出：“今天我需要去商店买冷饮、给汽车加油还有寄包裹”。假如有一个机器人（名字叫 DB2 优化器）负责决定如何完成这些任务，它可能会按照输入指令的顺序来完成任务，这样做当然可以完成任务，但很可能不是完成任务的最好方式。

如果先买冷饮，冷饮会不会在完成其他任务的过程中化掉？如果在去加油站的路上油就用光了怎么办？到达邮局的时候邮局会不会下班？

是的，机器人 DB2 需要做的是提供完成这些任务的最佳方案，然而至少目前它无法很好地完成这件事情，为什么？因为信息不足！需要提供足够的信息，比如商店、邮局、加油站的位置；商店、加油站、邮局的营业时间；冷饮可以保留多久不会化掉，现在车里有多少油？提供的信息越完善，越准确，越有可能得到最佳方案；否则，no 信息，no 优化！那么需要提供什么信息呢？想要 DB2 优化器制订最优方案，需要提供如下信息：

• 准确实时的统计信息；

- 合理配置的数据库配置参数；
- 正确的优化级别；
- 设计创建合理的索引；
- 磁盘 I/O 的合理设计(对应表空间的 transrate 和 overhead 参数)；
- 高效率运转的 SQL 语句。

为了在数据库中执行查询或 DML 语句(INSERT、UPDATE、DELETE)，DB2 必须创建一个访问计划(Access Plan)。访问计划定义按什么顺序访问表，使用哪些索引，以及用何种连接(Join)方法来关联数据。好的访问计划对于 SQL 语句的快速执行至关重要。DB2 优化器可以创建访问计划。这是一种基于成本的优化器，这意味着它是根据表和索引的相关统计信息来做出决策的。

本节主要讲解如下内容：

- DB2 优化器介绍；
- SQL 语句执行过程；
- 优化器组件和工作原理；
- 扫描方式；
- 连接方法；
- 优化级别；
- 如何影响优化器来提高性能。

2.4.2 DB2 优化器介绍

优化器是 DB2 的心脏和灵魂(可以把它类比成宝马 728 或波音 747 的发动机引擎)。它分析 SQL 语句并确定可以满足每条语句的最有效的存取路径(请参阅图 2.82 和图 2.83)。DB2 通过解析 SQL 语句来确定必须访问哪些表和列，从而完成该操作。然后 DB2 优化器查询存储在 DB2 系统目录中的系统信息和统计信息，以确定完成满足 SQL 请求所必需的任务的最佳方法。优化器在功能上等价于一个专家系统。专家系统是一个标准规则集合，当与情境数据组合时，它返回一个"专家"意见。例如，医学专家系统采用一个规则集合来确定哪些药可用于治疗哪些疾病，将规则集与描述疾病症状的数据组合，并将知识库应用于输入症状的列表。DB2 优化器会根据存储在 DB2 系统目录中的情境数据和 SQL 格式的查询输入来生成对数据检索方法的专家意见。

在数据库中优化数据访问是 DB2 最强大的功能之一(在所有数据库中，DB2 的优化器是最强大的)。用户在访问 DB2 数据时应告诉 DB2 要检索什么，而不是如何检索。不管数据是如何存储和操作的，DB2 和 SQL 都可以访问该数据。从物理存储特征中分离出的访问标准被称作物理数据独立性。DB2 优化器是完成该物理数据独立性的组件。

优化器根据许多信息执行复杂的计算。要使优化器的工作方式直观化，可以将优化器想象成一个执行以下步骤的过程：

(1) 接收并验证 SQL 语句的语法语义；

(2) 分析环境并优化满足 SQL 语句的方法；

(3) 创建计算机可读指令来执行优化的 SQL 语句；

(4) 执行指令或存储它们以便将来执行。

这个过程的第 2 步是最有趣的。优化器怎样决定如何以它的方式执行用户可以发送的大量 SQL 语句？优化器有许多类型的优化 SQL 的策略。它如何选择在优化存取路径中使用这些策略中的哪一个？IBM 并没有发布优化器如何确定最佳存取路径的真正和深入的详细信息，但优化器是一个基于成本的优化器(CBO)。这意味着优化器将始终尝试为每个查询制订减少总体成本的存取路径。要实现这个目标，DB2 优化器会应用查询成本公式，该公式对每条可能的存取路径的 4 个因素进行评估和权衡：CPU 成本、I/O 成本、DB2 系统目录中的统计信息和实际的 SQL 语句。

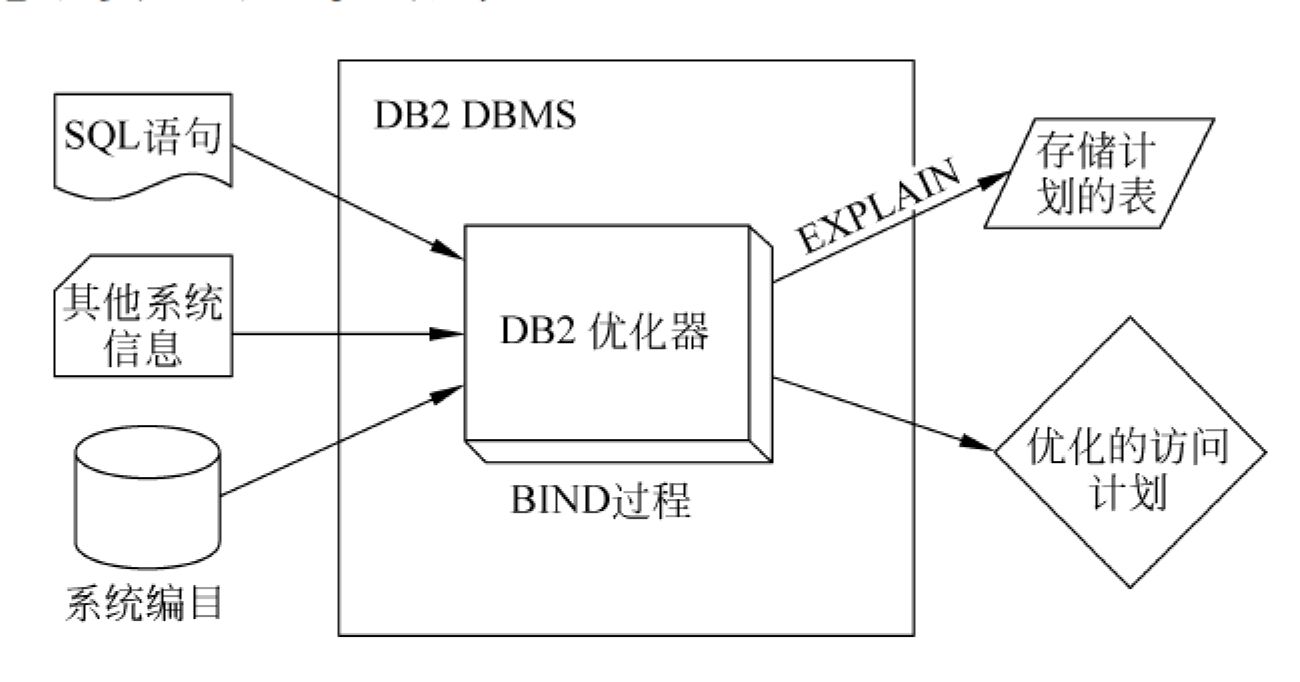


图 2.82　运行中的 DB2 优化器

2.4.3　SQL 语句执行过程

SQL 编译器需要执行几个步骤来产生可以执行的访问方案，如图 2.83 所示。注意，某些步骤仅针对联合数据库中的查询。

一条 SQL 语句的执行流程如下：

(1) 语法分析

SQL 编译器分析查询以验证其语法。如果检测到任何语法错误，那么查询编译器停止处理，并将适当的错误返回至提交该查询的应用程序。当解析完成时，创建该查询的内部表示，并将它存储在查询图模型中。

(2) 语义检查

编译器确保在语句的各个部分中没有不一致。作为语义检查的一个简单示例，编译器验证为 YEAR 标量函数指定的列的数据类型是日期时间数据类型。

编译器也将行为语义添加至该查询图模型，包括参考约束、表检查约束、触发器和视图的作用。该查询图模型包含查询的所有语义，包括查询块、子查询、相关、派生的表、表达式、数据类型、数据类型转换、代码页转换和分区键。

(3) 查询重写

编译器使用存储在查询图模型中的全局语义来将该查询变换为更易于优化的一种形式，并将结果存储在查询图模型中。

例如，编译器可能会移动谓词，改变其应用级别并有可能提高查询性能。这种类型的操作移动称为“一般谓词下推”。在一个分区数据库环境中，下列查询操作的计算更为集中：

- 聚集(avg，sum，max，min，count 等)；
- 行的重新分发；

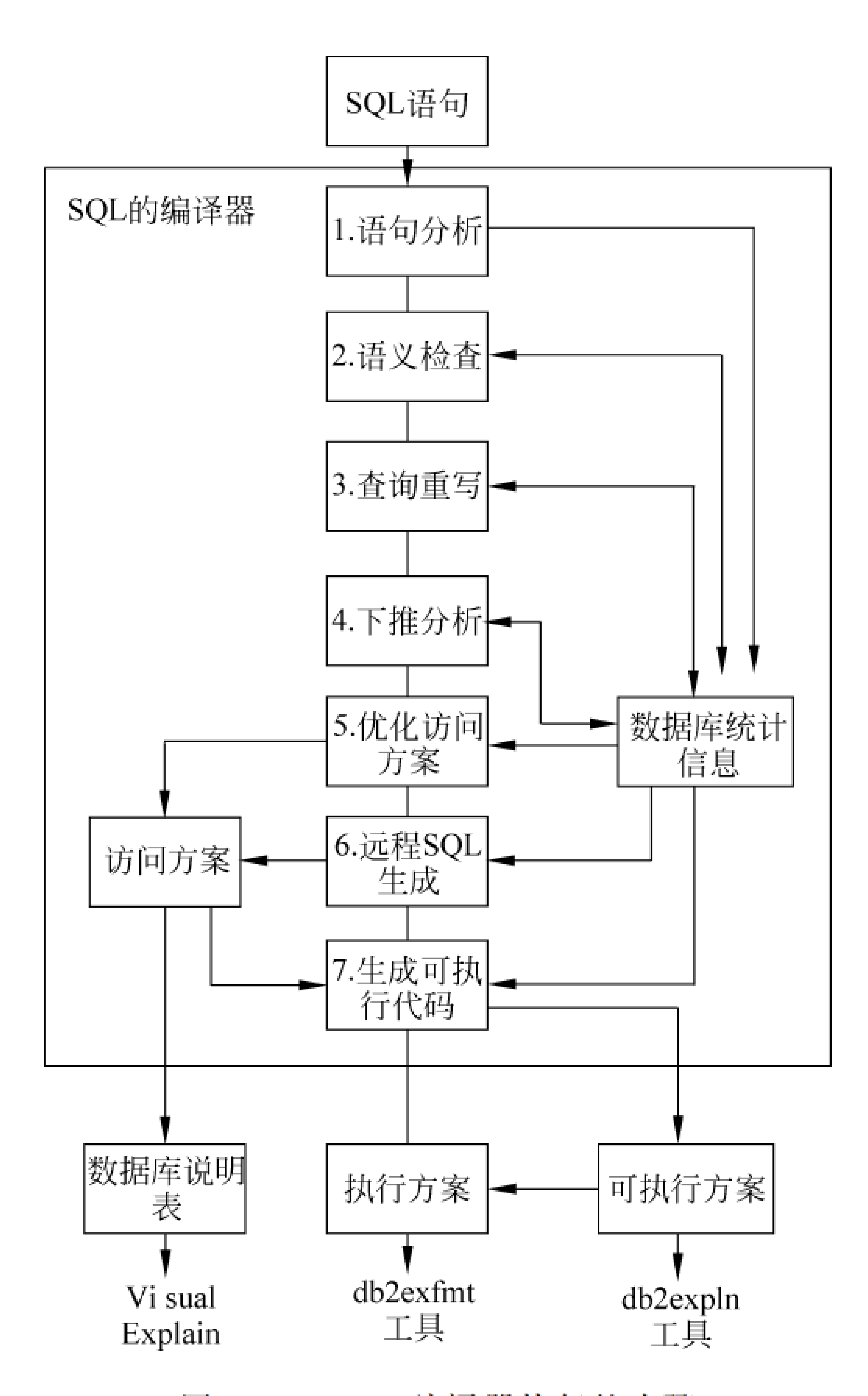


图 2.83　SQL 编译器执行的步骤

- 相关子查询，它是包含对该子查询之外表列的引用的子查询对于分区数据库环境中的某些查询，解除相关可能作为查询重写的一部分而发生。

(4) 下推分析

此步骤的主要任务是向优化器建议是否可在数据源处对一个操作以远程方式求值(即下推)。这种类型的下推活动仅适用于数据源查询，它是对一般谓词下推操作的扩展。

除非执行联合数据库查询，否则绕过此步骤。

(5) 优化访问方案

通过查询统计信息作为输入，编译器的优化器部分生成许多备用执行方案以满足查询。要估计每个备用方案的执行成本，优化器使用有关表、索引、列和函数的统计信息。然后，它选择具有最小估计执行成本的方案。优化器使用查询图模型来分析查询语义，并获取有关各种因素的信息，包括索引、基本表、派生的表、子查询、关联和递归。优化器还可考虑另一类下推操作，聚集(avg，sum，max，min，count 等)与排序，这种操作可将对这些操作求得的值下推到“数据管理服务”组件中来提高性能。在确定页大小选择时，优化器还考虑是否有不同大小的缓冲池。该环境是否包括一个分区数据库，以及是否能够为在对称多处理器(SMP)环境中实现分区内并行性改善选择的方案，均属考虑范围。优化器使用此信息来帮助选择该查询的最佳访问方案。编译器此步骤的输出是访问方案。此访问方案提供说明表

中捕获的信息。可以用说明快照捕获用于生成该访问方案的信息。

(6) 远程 SQL 生成

优化器选择的最终方案可由一组对远程数据源执行操作的步骤组成。对于每个数据源执行的那些操作，远程 SQL 生成步骤将根据数据源 SQL 语言创建一个高效的 SQL 语句。

(7) 生成可执行代码

在最后一个步骤中，编译器使用访问方案和查询图模型来创建查询的可执行访问方案或节(section)。此代码生成步骤使用查询图模型中的信息，以避免重复执行需要对一个查询只计算一次的表达式。可以进行此优化的示例包括代码页转换和绑定(宿主)变量的使用。

要对具有绑定(宿主)变量、专用寄存器或参数标记的静态和动态 SQL 语句启用查询(重新)优化，应将程序包与 REOPT 绑定选项进行绑定。如果使用此项，将使用绑定(宿主)变量、参数标记或专用寄存器的值，而不是由编译器选择的默认估计值，来优化属于该程序包并且包含这些变量的 SQL 语句的访问路径。当值可用时，执行查询时就会进行此优化。

有关静态 SQL 语句访问方案的信息存储在系统目录表(syscat. statements)中。当执行该程序包时，数据库管理器将使用存储在系统目录表中的信息来确定如何访问该数据，并提供该查询的结果。此信息由 db2expln 工具使用。

注意：在经常更改的表上以适当的时间间隔执行 RUNSTATS。优化器需要有关表及其数据的最新统计信息以创建最有效的访问方案。重新绑定应用程序以利用更新的统计信息。如果未执行 RUNSTATS 或优化器怀疑在空表或几乎是空的表上执行了 RUNSTATS，那么优化器可能使用默认值，或尝试根据在磁盘上存储该表所用的文件页的数目(FPAGES)，来得出特定的统计信息。已占用块的总数存储在 ACTIVE_BLOCKS 列中。

2.4.4 优化器组件和工作原理

理解 DB2 优化器的工作机制对于做性能调优是非常重要的。DB2 优化器主要由三种组件构成。

(1) Query Rewriter(查询重写)

SQL 是一个非常灵活的查询语言，通常会有很多不同的语句达到同一个目的。有时，优化器通过查询重写(Query Rewriter)会将一种形式的语句转换成另外一种形式，前提是第二种形式的语句执行起来更有效。

(2) Estimator(成本评估器)

成本评估产生三种度量标准：

- Selectivity：表示有多少 rows 可以通过谓词被选择出来，大小介于 0.0～1.0，0 表示没有 row 被选择出来。如果没有 statistics，Estimator 会使用一个默认的 selectivity 值，这个值根据谓词的不同而异。比如'='的 selectivity 小于'<'。如果有统计信息(Statistics)，比如对于 last_name='Smith'，Estimator 使用 last_name 列的 distinct 值的倒数(注：是指表中所有 last_name 的 distinct 值)作为 selectivity。
- Cardinality：表示一个 row set 的行数。
- Cost：Cost 表现了 Disk I/O、CPU usage 资源单位的使用成本单位(Timeron)。

(3) Plan Generator(计划生成器)

它的作用是尝试各种可能的执行计划,选择 cost 最低的一种。

2.4.5 扫描方式

查询优化器在编译一条 SQL 语句时会估计满足查询的不同方法的执行成本。根据它的估计,优化器选择优化访问方案。访问方案指定解析一条 SQL 语句所需的操作顺序。当绑定一个应用程序时,会创建一个程序包。此程序包包含该应用程序中所有静态 SQL 语句的访问方案。动态 SQL 语句的访问方案在执行应用程序时创建。

有两种方法来访问表中的数据:

- 顺序扫描整个表;
- 通过首先访问表上的索引来定位特定表行。

要产生查询请求的结果,根据谓词的条件选择行,通常在 WHERE 子句中说明这些条件。连接访问表中选择的行来产生结果集,并可以通过分组或排序输出来进一步处理结果集。

2.4.6 连接方法

连接是根据信息的某些公共域从两个或多个表组合信息的过程。当对应行中的信息符合连接条件时,一个表中的行就会与另一个表中的行配对。

在连接两个表时,无论使用哪种连接方法,总有一个表被选为外表(Outer table),而另一个表被选为内表(Inner table)。优化器根据所选连接方法的成本和类型决定哪个是外表、哪个是内表。首先访问外表,并且只扫描一次。根据连接的类型和存在的索引,可以多次扫描内表。还有一点也很重要,即使用户试图连接两个以上的表,优化器也将每次只连接两个表,并在必要时保存中间结果。

例如,考虑下面的表 2.1 的 Table1 和 Table2:

表 2.1 Table1 和 Table2 表

Table1		Table2	
PROJ	PROJ_ID	PROJ_ID	名称
A	1	1	Sam
B	2	3	Joe
C	3	4	Mary
D	4	1	Sue
		2	Mike

要将标识列具有相同值的 Table1 和 Table2 连接,使用下列 SQL 语句:

```
SELECT PROJ,x.PROJ_ID,NAME
FROM TABLE1 x,TABLE2 y
WHERE x.PROJ_ID = y.PROJ_ID
```

此查询得到下列一组结果,如表 2.2 所示。

表 2.2 查询结果

PROJ	PROJ_ID	PROJ_ID
A	1	Sam
A	1	Sue
B	2	Mike
C	3	Joe
D	4	Mary

根据连接谓词是否存在，以及所涉及的由表和索引统计信息来确定的各种成本，优化器选择下列其中一种连接方法：

- 嵌套循环连接；
- 合并连接；
- 哈希连接。

可以提供显式连接运算符(如 INNER 或 LEFT OUTER JOIN)来确定如何在连接中使用表。但是，以这种方式改变查询之前，应允许优化器确定如何连接表。然后分析查询性能来决定是否要添加连接运算符。

当查询需要连接表时，优化器可以选择前面所说的三种基本连接策略的其中一种。

正如大家所知，初看起来，某些方法与其他方法相比是更好的选择。例如，与根据外表的每一行扫描内表的嵌套循环连接相比，合并连接具有只对表扫描一次的优势。于是，合并连接似乎是一个更好的选择。但是，如果存在索引的话，则嵌套循环连接会是更好的选择。同样地，与合并连接相比，哈希连接似乎是更好的选择，因为它不需要在执行前对输入表排序，但如果需要保持外表中行的次序，则合并连接或嵌套循环连接可能是更好的选择，但是哈希连接不能保证维持次序，因为它可能溢出到磁盘以致破坏次序。

那么 DB2 优化器如何针对特定连接来决定使用哪种连接方法呢？首先，它必须考虑查询中谓词的类型。当选择了可能的连接方法时，DB2 优化器随后根据成本模型和选定的优化级别来决定使用哪种连接方法。优化级别是数据库配置文件中可配置的参数，它告诉优化器要进行多大程度的优化。这个值越高，优化操作就越多。优化级别可能的值为：0、1、2、3、5、7 和 9。这些值对可能的连接方法的影响如下：

- 嵌套循环连接在每个优化级别都可行；
- 合并连接在优化级别 1 及以上级别是可行的；
- 哈希连接在优化级别 5 及以上级别是可行的。

DB2 优化器会根据成本模型来决定使用合适的连接。所以只需要提供给优化器成本模型必需的信息：准确的统计信息、合理的配置参数、高效的 SQL 和索引，以及优化级别。下一节将讲优化级别。

2.4.7 优化级别

优化级别指定了各种优化策略，当编译和优化 SQL 语句时，优化器将使用这些策略。连接方法的选择就取决于正在使用的优化级别。所以，优化器并非总是使用上面所描述的每种存取路径技术。相反，根据优化级别，优化器使用各种不同的技术。优化级别的用途是通过它来指导 DB2 何时采用哪种优化策略和优化技术。通常，优化器考虑的优化策略越

多，用于查询的存取方案就越好。然而，当优化器被指导去考虑的优化策略越多，把 SQL 编译成可执行的存取路径的准备时间就越长。幸运的是，可以设置优化级别来限制优化查询时所应用的优化技术数目。对于较简单的查询、资源受限系统和动态 SQL，这是非常有用的。

影响为 SQL 语句生成何种访问方式的最重要因素就是优化级别，优化级别用于为此任务做准备。该信息告诉 DB2 优化器要付出多少努力，使用什么优化技术来确定解决查询的最佳访问计划。较高的级别将使优化器使用更为复杂的算法和代数分析，因而也需要花费更多的准备时间，来生成最终的访问计划。

有 7 个可用的优化级别，各级别使用所有可用规则和统计信息的不同子集。可用的优化级别包括：

0——使用最少的优化

1——所用的优化程度大致与 DB2/6000 V1 相同，此外还增加了一些 V1 中没有的低成本特性

2——使用优化级别 5 的特性，但简化了连接算法

3——执行中等数量的优化，与 DB2 for MVS/ESA 的查询优化特征相似

5——使用大量优化，利用 Heuristic Rules（若未另行指定，这将是默认优化级别）

7——使用无 Heuristic Rules 的大量优化

9——使用所有可用优化技术

提示：

Heuristic Rules 其实就是 DB2 提供了一种人工智能的专家系统，它存储了各种优化规则来为 SQL 生成最优的访问计划。

在确定要使用的最佳优化级别时，以下指导原则可给予用户很大的帮助：

- 为需要很少的优化、极其依赖主键索引搜索或极简单的连接（例如，非常简单的 OLTP）的查询使用优化级别 0 或 1；
- 若查询比较简单，仅包含少量表和设计表的索引的连接（例如，OLTP），则设查询使用优化级别为 1；
- 对包含复杂 OLTP 的工作负载或包含多个表上许多复杂连接的报告（例如，混合的 OLTP 和报告），应设使用优化级别为 5；
- 对于需要大量数据统计数据分析且运行时间较长（超过 1 分钟）的查询（例如，非常复杂的数据研究或决策支持），使用优化级别设为 9。DB2 优化器需要花费较长的时间来生成访问计划，但访问计划中改进的成本往往超出产生计划所花费的额外时间。

2.4.8 如何影响优化器来提高性能

读者已经理解了优化器的工作原理和组件，优化器能否正常高效工作取决于以下影响优化器工作的重要性能准则。因此，只要记住这些影响 DB2 优化器的准则，就可以实现这些准则以便获得更好的 SQL 性能。

（1）使 DB2 统计信息保持最新

如果没有存储在 DB2 系统目录中的统计信息，优化器在优化任何事务时都会遇到困

难。这些统计信息向优化器提供了与正在被优化的 SQL 语句将要访问的表状态相关的信息。存储在系统目录中的统计信息的类型包括：

- 关于表的信息，包括总的行数、关于压缩的信息和总页数；
- 关于列的信息，包括列的离散值的数量和存储在列中的值的分布范围；
- 关于表空间的信息，包括活动页面的数量和硬件描述（Transtrate 和 Overhead）；
- 索引的当前状态，包括是否存在索引、索引的组织（叶子页的数量和级别的数量）、索引键的离散值数量，以及是否群集索引；
- 关于表空间和索引节点组或分区的信息。

（2）构建适当的索引

为保证最佳 DB2 应用程序性能而可以做的最重要的事就是根据应用程序使用的查询，为表创建正确的索引。创建的索引不是多多益善，而是要遵循一些创建索引的原则。

（3）配置合理的数据库配置参数

优化器在工作时，它需要读取相关数据库配置参数，这些配置参数如下：

- AVG_APPLS（平均应用程序）：此参数表示为数据库并发运行应用程序的平均数量。DB2 使用此信息来确定排序空间和缓冲池使用得有多么频繁，并确定查询能够使用的空间有多少。
- SORTHEAP（排序堆）：排序堆是执行排序时可用的内存空间数量。若排序需要的内存多于排序堆中可用内存，则部分排序数据将不得不分页到磁盘上（这会对性能造成严重的负面影响）。
- LOCKLIST（锁列表）：该参数表示 DB2 可用于存储各应用程序的锁定信息的内存数量。若锁列表空间过小，则 DB2 可能必须逐步升级（Escalate）部分锁，以便为应用程序具有的所有锁腾出空间。
- MAXLOCKS（最大锁列表百分比）：该参数控制整个锁列表空间中有百分之多少的空间可为一个应用程序所有。若一个应用程序具有过多的开放锁，从而试图占用过多的内存，DB2 将升级部分锁，以释放锁列表中的空间，这会影响并发性能。
- NUM_FREQVALUES（频繁值数）：DB2 Runstats 实用工具使用频繁值数来控制 DB2 将在内存中保留多少使用频率最高的值。优化器使用该信息来确定 WHERE 子句中的一个谓词将消耗结果集的多少百分比。
- NUM_QUANTILES（数据分位数）：DB2 Runstats 实用工具使用分位数来控制为列数据捕获多少分位。增加分位数将给予 DB2 关于数据库中数据分布情况的更多信息。
- DBHEAP（数据库堆）：数据库堆控制数据库对象信息的可用内存量。对象包括索引、缓冲池和表空间。事件监控器和日志缓冲区信息也存储在这里。
- CPUSPEED（CPU 速度）：计算机的 CPU 速度。
- BUFFPAGE 和缓冲池大小：优化器可在优化数据中使用的缓冲池大小。增加或减少缓冲池大小会对访问计划产生显著影响。

（4）选择合适的优化级别

用于指定在编译 SQL 查询时所使用的默认优化级别。对于混合的 OLTP/OLAP，使用 5 或 3 作为默认值；对于 OLTP，使用一个更低的级别；而对于 OLAP，则使用一个更高

的级别。对于简单的 SELECT 或短的运行时查询(通常只需花不到 1 秒就可以完成),使用 1 或 0 也许比较合适。如果有很多的表,有很多相同列上的连接谓词,那么尝试级别 1 或 2。对于超过 30 秒才能完成的长时间运行的查询,或者如果要插入一个 UNION ALL VIEW (这是在 FixPak4 中加进来的),那么可以尝试使用级别 7。在大多数环境下都应该避免使用级别 9。用户一定要根据自己的业务类型和特点来选择合适的优化级别。

(5) 合理的存储 I/O 设计

优化器在工作时,需要读取相关硬件信息,这些信息是通过表空间的 Transrate 和 Overhead 这两个参数体现的。

- Transrate=(1/传送速率) * 1000/1024000 * 4096(假设用 4KB 页大小);
- Overhead=平均寻道时间+(((1/磁盘转速) * 60 * 1000)/2)而平均寻道时间、磁盘转速和传送速率是由硬盘本身决定的。所以必须做合理的存储 I/O 设计和选择转速更快的硬盘以影响优化器更好地工作。

(6) 良好的应用程序设计和编码

进行合适的应用编程和写出高效的 SQL。

2.5　SQL 调优概述

经常听到有做应用的朋友抱怨数据库的性能问题,比如非常低的并发,令人崩溃的响应时间,长时间的锁等待,锁升级,甚至是死锁,等等。在解决这些问题的过程中,DBA 经常发现应用开发人员对数据库的“误用”。包括,返回过多不必要的数据,不必要和不适当加锁,对隔离级别的误用和对存储过程的误用等等。但是,面对浩如烟海的数据库知识,要求完全掌握,对应用开发人员来说也确实枯燥艰深。因此,笔者特别提炼对应用开发人员有帮助的 SQL 书写部分,以期望能对数据库开发人员有所帮助。

本节将主要讨论基于语法的优化以及简单的查询条件。基于语法的优化指的是不考虑任何的非语法因素(例如,索引,表大小和存储等),仅考虑在 SQL 语句中对于词语的选择以及书写的顺序。

2.5.1　一般规则

这一部分,介绍一些在书写简单查询语时需要注意的通用的规则。

(1) 根据权值来优化查询条件

最好的查询语句是将简单的比较操作作用于最少的行上。以下两张表,表 2.3 和表 2.4 以由好到差的顺序列出了典型查询条件操作符并赋予权值。

表 2.3　查询条件中操作符的权值

操　作　符	权　　值	操　作　符	权　　值
=	10	>	5
>=	5	<	5
<=	5	LIKE	3
<>	0		

表 2.4 查询条件中操作数的权值

操 作 数	权 值
仅常量字符	10
仅有列名	5
仅有参数	5
多操作数表达式	3
精确数值类型	2
其他数值类型	1
时间数据类型	1
字符数据类型	0
NULL	0

根据表 2.3 和表 2.4 中分配的权值，可以看出最好的查询条件应该是像下面这样的：

```
… WHERE smallint_column = 789
```

这个例子得到 27 分，计算如下：

- 左侧只有列名(smallint_column)得 5 分；
- 操作数为精确数据类型(smallint_column)得 2 分；
- 等号(=)操作符得 10 分；
- 右侧是文字字符(789)得 10 分。

下面是另外一个例子

```
... WHERE char_column >= varchar_column || 'x'
```

这种类型的查询权值得分就很低，只有 13 分：

- 左侧只有列名(char_column)得 5 分；
- 字符数据类型的操作数得 0 分；
- 大于等于操作符得 5 分；
- 左侧是多操作数表达式得 3 分；
- VARCHAR 类型的操作数得 0 分。

上面表格中的权值数可能在不同类型的数据库系统中会有所不同，所以记住这些具体数值是没有意义的，只需要了解它们的排序即可。用时越少的比较条件，得分也就越高，这样的比较条件通常是那些操作的行数少或者易于比较的。

(2) 传递法则

传递法则是这样定义的：

```
IF
(A < comparison operator > B) IS TRUE
AND (B < comparison operator > C) IS TRUE
THEN
(A < comparison operator > C) IS TRUE
AND NOT (A < comparison operator > C) IS FALSE
```

比较运算符包括：=、>、>=、<、<=、+，但不包括：<>、LIKE。

通过传递法则，可以看出，可以用 C 来替换 B，而不使表达式的意思发生变化。

下面的两个例子表达了同样的含义，但是表达式2要比表达式1执行的快。

表达式1：

```
... WHERE column1 < column2
AND column2 = column3
AND column1 = 5
```

表达式2：

```
... WHERE 5 < column2
AND column2 = column3
AND column1 = 5
```

大多数的数据库管理系统都会自动做这样的调整，但是当表达式中含有括号时，它们就不会自动调整了。例如一个如下的SELECT语句：

```
SELECT * FROM Table1
WHERE column1 = 5 AND
NOT (column3 = 7 OR column1 = column2)
```

如果进行转化的话，会得到如下的语句：

```
SELECT * FROM Table1
WHERE column1 = 5
AND column3 <> 7
AND column2 <> 5
```

进行这样变化后的语句会比第一个执行的更快。

(3) Sargability

理想的SQL表达式应该采用下面这种通用的格式：

```
<column><comparison operator><literal>
```

IBM研究人员将这种查询条件语句命名为Srgable Predicates，因为SARG是Search ARGument的组合。

根据这一规则，查询条件的左侧应该是一个列名；右侧应该是一个很容易进行查找的值。

遵循这一规则，所有的数据库系统都会将如下的表达式：

```
5 = column1
```

转换成：

```
column1 = 5
```

但是当查询条件中包含算术表达式时，只有部分的数据库系统进行转换。

例如：

```
... WHERE column1 - 3 = - column2
```

转换成：

```
... WHERE column1 = - column2 + 3
```

还是可以带来查询性能的优化的。

2.5.2 针对专门操作符的调优

前面,讲的是关于查询条件的一般规则,在这一节中,将讨论如何使用专门的操作符来改进 SQL 代码的性能。

(1) 与(AND)

数据库系统按从左到右的顺序来解析一个由 AND 连接的表达式,但是 Oracle 却是个例外,它是从右向左地解析表达式。可以利用数据库系统的这一特性,来将概率小的表达式放在前面,或者是如果两个表达式可能性相同,那么可将相对不复杂的表达式放在前面。这样做的话,如果第一个表达式为假的话,那么数据库系统就不必再费力去解析第二个表达式了。例如,可将

```
... WHERE column1 = 'A' AND column2 = 'B'
```

转换成:

```
... WHERE column2 = 'B' AND column1 = 'A'
```

这里假设 column2='B'的概率较低,如果是 Oracle 数据库的话,只需将规则反过来用即可。

(2) 或(OR)

和与(AND)操作符相反,用或(OR)操作符写 SQL 语句时,就应该将概率大的表达式放在左面,因为如果第一个表达式为假的话,OR 操作符意味着需要进行下一个表达式的解析。

(3) 与+或

按照集合的展开法则,

A AND (B OR C)与(A AND B) OR (A AND C)是等价表达式。

假设表 2.5 中,要执行一个 AND 操作符在前的表达式:

```
SELECT * FROM Table1
WHERE (column1 = 1 AND column2 = 'A')
OR (column1 = 1 AND column2 = 'B')
```

表 2.5 AND+OR 查询

Row#	Colmun1	Column2
1	3	A
2	2	B
3	1	C

当数据库系统按照查询语进行搜索时,它按照下面的步骤执行:

- 索引查找 column1=1,结果集={row 3};
- 索引查找 column2='A',结果集={row1};
- AND 合并结果集,结果集={};
- 索引查找 column 1=1,结果集={row 3};

- 索引查找 column 2＝'B'，结果集＝{row2}；
- AND 合并结果集，结果集＝{}；
- OR 合并结果集，结果集＝{}。

现在根据集合的展开法则，对上面的语句进行转换：

```
SELECT * FROM Table1
WHERE column1 = 1
AND (column2 = 'A' OR column2 = 'B')
```

按照新的顺序进行查询搜索时，它按照下面的步骤执行：

- 索引查找 column2＝'A'，结果集＝{row1}；
- 索引查找 column 2＝'B'，结果集＝{row2}；
- OR 合并结果集，结果集＝{}；
- 索引查找 column1＝1，结果集＝{row 3}；
- AND 合并结果集，结果集＝{}。

由此可见搜索次数少了一次。虽然一些数据库操作系统会自动进行这样的转换，但是对于简单的查询来说，这样的转换还是有好处的。

(4) 非(NOT)

让非(NOT)表达式转换成更易读的形式。简单的条件能通过将比较操作符进行反转来达到转换的目的，例如：

```
... WHERE NOT (column1 > 5)
```

转换成：

```
... WHERE column1 <= 5
```

比较复杂的情况，根据集合的摩根定理：

```
NOT (A AND B) = (NOT A) OR (NOT B)和 NOT (A OR B) = (NOT A) AND (NOT B)
```

根据这一定理，可以看出它至少有两次的搜索有可能减少为一次。如下的查询条件：

```
... WHERE NOT (column1 > 5 OR column2 = 7)
```

可以转换成：

```
... WHERE column1 <= 5
AND column2 <> 7
```

但是，当转换成后的表达式中有不等操作符<>，那么性能就会下降，毕竟，在一个值平均分布的集合中，不等的值的个数要远远大于相等的值的个数，正因为如此，一些数据库系统不会对非比较进行索引搜索，但是它们会为大于或小于进行索引搜索，所以可以将下面的查询进行如下转换：

```
... WHERE NOT (column1 = 0)
```

转换成：

```
... WHERE column < 0
```

```
OR column > 0
```

(5) IN

很多人认为如下的两个查询条件没有什么差别,因为它们返回的结果集是相同的。

条件 1:

```
... WHERE column1 = 5
OR column1 = 6
```

条件 2:

```
... WHERE column1 IN (5,6)
```

这样的想法并不完全正确,对于大多数的数据库操作系统来说,IN 要比 OR 执行得快。所以如果可以的话,要将 OR 换成 IN。

当 IN 操作符,是一系列密集的整型数字时,最好是查找哪些值不符合条件,而不是查找哪些值符合条件,因此,如下的查询条件就应该进行如下的转换:

```
... WHERE column1 IN (1,3,4,5)
```

转换成:

```
... WHERE column1 BETWEEN 1 AND 5
AND column1 <> 2
```

当一系列的离散的值转换成算数表达式时,也可获得同样的性能提高。

(6) UNION

在 SQL 中,两个表的 UNION 就是两个表中不重复的值的集合,即 UNION 操作符返回的两个或多个查询结果中不重复行的集合。这是一个很好的合并数据的方法,但是这并不是最好的方法。

查询 1:

```
SELECT * FROM Table1
WHERE column1 = 5
UNION
SELECT * FROM Table1
WHERE column2 = 5
```

查询 2:

```
SELECT DISTINCT * FROM Table1
WHERE column1 = 5
OR column2 = 5
```

在上面的例子中,column1 和 column2 都没有索引。如果查询 2 总是比查询 1 执行快的话,那么就可以建议总是将查询 1 转换成查询 2,但是有一种情况,这样做在一些数据库系统中可能会带来性能变差,这是由于两个优化缺陷所造成的。

第一个优化缺陷就是很多优化器只优化一个 SELECT 语句中一个 WHERE 语句,所以查询 1 的两个 SELECT 语句都被执行。首先优化器根据查询条件 column1=5 为真来查找所有符合条件的所有行,然后根据查询条件 column2=5 为真来查找所有符合条件的所

有行，即两次表扫描，因此，如果 column1=5 没有索引的话，查询 1 将需要 2 倍于查询 2 所需的时间。如果 column1=5 有索引的话，仍然需要二次扫描，但是只有在某些数据库系统存在一个不常见的优化缺陷却将第一个优化缺陷给弥补了。当一些优化器发现查询中存在 OR 操作符时，就不使用索引查询，所以在这种情况下，并且只有在这种情况下，UNION 才比 OR 性能更高。这种情况很少见，所以仍然建议大家当待查询的列没有索引时使用 OR 来代替 UNION。



第3章

数据库开发

本章将会讨论数据库开发的问题，讲解 DB2 与 JDBC 支持，旧的 JDBC 驱动程序与新的通用 JDBC 驱动程序的比较等内容。

3.1 DB2 与 JDBC 支持

依照 JDBC 规范，有四种类型的 JDBC 驱动程序体系结构：

Type 1：这类驱动程序将 JDBC API 作为到另一个数据访问 API 的映射来实现，如开放式数据库连通性（Open Database Connectivity，ODBC）。这类驱动程序通常依赖本机库，这限制了其可移植性。JDBC-ODBC 桥驱动程序就是 Type 1 驱动程序的最常见的例子。

Type 2：这类驱动程序部分用 Java 编程语言编写，部分用本机代码编写。这些驱动程序使用特定于所连接数据源的本机客户端库。同样，由于使用本机代码，所以其可移植性受到限制。

Type 3：这类驱动程序使用纯 Java 客户机，并使用独立于数据库的协议与中间件服务器通信，然后中间件服务器将客户机请求传给数据源。

Type 4：这类驱动程序是纯 Java，实现针对特定数据源的网络协议。客户机直接连接至数据源。

对于 DB2 UDB V7.2 来说，它不支持 Type 1 和 Type 4 的驱动程序，但是提供了分别支持 Type 2 和 Type 3 的驱动程序。

示例如下：

两种驱动程序均随产品安装由 db2java.zip 提供。

（1）COM.ibm.db2.jdbc.app.DB2Driver

这是一种 Type 2 的 JDBC 驱动程序，它通过 DB2 本地客户机库的帮助建立和 DB2 本地数据库或是远程数据库的连接（事先将远程数据库编目到本地）。因此，必须在应用系统所在的机器上同时部署 DB2 本地客户机库，这也许是它最大的一个不足之处。

使用格式如下：

```
Driver Name: COM.ibm.db2.jdbc.app.DB2Driver
```

```
URL Pattern: jdbc:db2:databasename
databasename: 需要访问的数据库名
```

（2）COM.ibm.db2.jdbc.net.DB2Driver

这是一种 Type 3 的 JDBC 驱动程序，它通过与一台已经部署了 DB2 本地客户机库的机器通信来建立和 DB2 远程数据库的连接。

使用格式如下：

```
Driver Name: COM.ibm.db2.jdbc.net.DB2Driver
URL Pattern: jdbc:db2:ServerIP:databasename
ServerIP: 需要访问的数据库所在机器 IP 地址
databasename: 需要访问的数据库名
```

（目标 DB2 系统侦听该服务于默认端口 6789，否则还需要在 URL Pattern 中指定目标端口号）

对于 DB2 UDB V8.1 来说，它仍然不支持 Type 1 的驱动程序。同时，它在 DB2 UDB V8.1 的基础上，新增加了对 Type 4 驱动程序的支持。

示例如下：

DB2 UDB V8.1 仍然支持上面所述 V7.2 支持的两种驱动程序，随产品安装由 db2java.zip 提供，但具体实现上和 DB2 UDB V7.2 产品发布的包有所不同，所以可能存在下文所要进行实验验证的兼容性问题。

除了 COM.ibm.db2.jdbc.app.DB2Driver 之外，DB2 UDB V8.1 还提供了另外一种 Type 2 的驱动程序，随产品安装由 db2jcc.jar 提供。其实现包名是 com.ibm.db2.jcc.DB2Driver，在 DB2 UDB V8.1 最初的实现中，此驱动程序只用于使用 Type 4 驱动程序体系结构与 DB2 服务器进行直接的 Java 连接，这类驱动程序由于不需要另外部署 DB2 本地客户机库以及性能相对较好而收到开发人员的欢迎。自从 DB2 UDB V8.1.2（安装了 FixPack 2）之后，开发人员还可以在 Type 2 体系结构中使用该驱动程序，以提高本地应用程序的性能。

这里，两种驱动程序具有相同的实现类名称，有两种不同的方法可以区分 DB2 系统在内部最终会实例化哪个驱动程序：

（1）使用不同的 URL Pattern 来区分两种不同的驱动程序。

```
Type 2 Driver URL Pattern: jdbc:db2:databasename
```

这当中 databasename 是需要访问的数据库名

```
Type 4 Driver URL Pattern: jdbc:db2://ServerIP:50000/databasename
```

这当中 ServerIP 是需要访问的数据库所在机器 IP 地址，databasename 是需要访问的数据库名，DB2 服务器会在默认端口 50000 上进行侦听。

（2）使用连接特性来区分数据库连接是否会使用 DB2 本地客户机库，或者是使用 Java 直接连接。

DB2 UDB V8.1 新增加支持的这种 Type 4 驱动程序，常被称为“通用 JDBC 驱动程序”，是一种与驱动程序类型连通性或目标平台无关的抽象 JDBC 处理器，因此常用于进行分布式和本地 DB2 UDB 访问。因为“通用 JDBC 驱动程序”独立于任何特定 JDBC 驱动程

序类型连通性或目标平台，所以它在一个 DB2 UDB 驱动程序实例中同时支持所有 Java 连通性（Type 4 驱动程序）和基于 JNI 的连通性（Type 2 驱动程序）。该驱动程序可以用于独立 Java 应用程序或多层应用程序，是开发人员一个不错的选择。

3.2 理解 DB2 UDB JDBC 通用驱动程序

在 DB2 环境中的 Java 开发的演变过程中，最近的动向是 DB2 UDB JDBC 通用驱动程序。这种新的驱动程序提供了很多优点和改进，使它成为应用程序开发的最佳选择。这节读者将理解这种驱动程序的内部工作原理，并看看它是怎样匹配整个应用程序开发计划的。

首先比较两种现有的驱动程序：

- 旧的 CLI 驱动程序；
- 新的 JDBC 通用驱动程序。

在本节中，主要通过以下几方面来展示这两种驱动程序之间的不同之处：

- 安装；
- 连接；
- 驱动程序初始化；
- 特性；
- 错误处理；
- 事务管理。

并且本节将讨论问题诊断和对跟踪的分析。要理解如何做这件事，需要了解 SQLException，以及它与 JDBC 有怎样的关联。对于新的 JDBC 通用驱动程序，将讲解如何进行 JCC 跟踪，以及进行 JCC 跟踪时需要些什么。完成跟踪后，将深入了解跟踪由哪些部分组成，以及如何使用跟踪来帮助找到问题的根源。

3.2.1 旧的 JDBC 驱动程序与新的通用 JDBC 驱动程序的比较

要想理解对 DB2 通用驱动程序的开发，就需要理解 JDBC 规范如何定义用于 Java 的不同类型的驱动程序。

(1) Type 1 驱动程序

这类驱动程序的代码直接与本机 API 形成映射。JDBC 和 ODBC 是类似的 API，所以这种驱动程序常常与 JDBC-ODBC 桥联系在一起。

这类驱动程序与 DB2 UDB 产品没有太多的关联。

(2) Type 2 驱动程序

Type 2 驱动程序中有一个本机组件，该组件是驱动程序的一部分，但与数据访问 API 相分离。这个本机组件和 Java 组件一起构成驱动程序。

对于 DB2 UDB，DB2 CLI 库包含本机组件。

(3) Type 3 驱动程序

这是一个 Java 客户机，使用独立于数据库的协议进行通信。

由于这种协议是独立于数据库的，这个优点使之适合于作为异构后端服务器网关的中间件服务器。

(4) Type 4 驱动程序

这类驱动程序是纯 Java 编写的，它实现了用于特定数据源的网络协议。客户机直接连接到数据源。

谈到 DB2 UDB，用户只需关心 Type 2、3 和 4 驱动程序。有了前面介绍的知识，现在可以看看关于 Type 2 和 Type 4 驱动程序的一些专门信息，并考察在应用程序开发中使用 Type 4 驱动程序的优点。来看看旧的 CLI Type 2 驱动程序与 Type 4 通用 JDBC 驱动程序之间的比较。

- 安装

DB2 JDBC 支持包含在 DB2 UDB 客户机和服务器的 Java enablement 选项中。不需要专门安装 DB2 JDBC 驱动程序，用户只需确保下载了适合于平台的 Java 开发工具箱(JDK)。DB2 Information Center 包含关于如何在 UNIX 和 Windows 上为 Java 设置环境的详细信息。表 3.1 所示是旧的 CLI 驱动程序和通用驱动程序的安装比较。

表 3.1　安装比较

旧的 CLI 驱动程序	通用驱动程序
旧的 CLI 驱动程序的物理表示是 db2java.zip 文件	通用 JDBC 驱动程序的物理表示是 db2jcc.jar 文件
在 UNIX 环境中，只需在 CLASSPATH 中有 sqllib/java/db2java.zip，就可以使用旧的 Type 2 驱动程序。在 Windows 上也是如此	在 UNIX 环境中，只需在 CLASSPATH 中有 db2jcc_license_cu.jar 和 sqllib/java/db2jcc.jar，就可以使用 Type 4 通用驱动程序。在 Windows 上也是如此
支持这类驱动程序的有 JDBC 2.0 和部分 JDBC 3.0	支持这类驱动程序的是大多数 JDBC 3.0 实现，只要安装了 JDK1.4.x 作为 Java 包的一部分，就提供了对这类驱动程序的支持

- 连接

这两类驱动程序的不同之处表现在它们建立连接的方式上。JDBC 的基本功能是连接到数据库，并发送 SQL 语句到服务器。它能够处理结果集，并将其发送给请求者。表 3.2 所示是旧的 CLI 驱动程序和通用驱动程序的连接比较。

表 3.2　连接比较

旧的 CLI 驱动程序	通用驱动程序
到数据库的连接是通过一个本机数据库接口进行的。在这里，DB2 使用 CLI。JDBC 层位于 CLI 之上，CLI 是与数据库服务器通信的本机组件	一切都是纯 Java 的，与数据库的通信通过网络通信完成。DB2 UDB 使用分布式关系数据库架构(DRDA)来与服务器进行通信，并将请求传递给数据库服务器
因为旧的 CLI 驱动程序需要公共客户机代码，所以它还需要一个 DLL/共享对象。为了使用这类驱动程序，必须安装 DB2 产品	这是一种纯 Java 的驱动程序，可独立于它所在机器上安装的产品而运行。也就是说，可以将它看作一个单独的实体，它是独立于附带它的那个 DB2 产品的

- 驱动程序初始化

在使用不同的驱动程序时，用于装载该驱动程序的代码也会有所不同。有两种建立连接的方式。和所有 JDBC 资源一样，在使用完连接时，要调用连接关闭方法。表 3.3 所示是

旧的 CLI 驱动程序和通用驱动程序的驱动程序初始化连接比较。

表 3.3 驱动程序初始化比较

旧的 CLI 驱动程序	通用驱动程序
为装载驱动程序和建立连接,需要三个基本步骤: 导入 JDBC 核心类(例如 import java. sql *) 装载 JDBC 驱动程序 Class. forName (COM. ibm. db2. jdbc. app. DB2Driver) 指定连接 URL:DriverManager getConnection jdbc:db2:coffebk	通用驱动程序支持通过单个 driver. h 网络通信的 Type 2 和 Type 4 连接。DB2 UDB 使用分布式关系数据库架构(DRDA)来与服务器进行通信,并将请求传递给数据库服务器。 所使用的是 Type 2 还是 Type 4 驱动程序,是通过连接的形式来指定的。下面的连接形式表明使用的是 Type 2 还是 Type 4 驱动程序: jdbc:db2//server:port/database jdbc:db2//server/database 下面的连接形式意味着使用的驱动程序是 Type 2 驱动程序: jdbc:db2:database
如果愿意,也可以使用 Type 3 驱动程序,在这种情况下,驱动程序的初始化就是:COM. ibm. db2. jdbc. net. DB2Driver	可以通过所使用的连接级别在这两类驱动程序的物质层之间进行切换

• 特性

随着 DB2 UDB,Version 8 的出现,Java 开发变得更加强大,而且编程工作也更具有独立性。现在,开发过程中的大部分精力都集中在添加新特性、改善新的 JDBC 通用驱动程序的内存管理和稳定性上。表 3.4 所示是旧的 CLI 驱动程序和通用驱动程序的特性比较。

表 3.4 特性比较

旧的 CLI 驱动程序	通用驱动程序
这类驱动程序需要专门安装 DB2 UDB 产品,因为它依赖于该产品的本机代码	这类驱动程序可以看作一个独立的产品。不需要安装产品,因为可以随很多 DB2 平台一起提供它
旧的驱动程序版本是和 DB2 UDB 修复包相对应的,只能随一个修复包一起发布	JCC 驱动程序的发布独立于修复包。JCC 驱动程序有它们自己的版本,它们是根据 DB2 产品不同发行版的需要发布的。例如,DB2 V8. 20 fp9 可能附带 2. 3. 9 版的 JCC 驱动程序,而 DB2 V8. 20 OS/390 PTF UQ72081 可能附带 2. 3. 11 版的 JCC 驱动程序

• 错误处理

这两类 JDBC 驱动程序处理错误的方式大相径庭。新驱动程序的错误消息仍在开发中,但是对于通用驱动程序而言,版本越新,其错误处理功能越好。典型的 JDBC 异常通常由 SQLErrorCode、SQLState 和 SQLMessage 组成。表 3.5 所示是旧的 CLI 驱动程序和通用驱动程序错误处理比较。

表 3.5 错误处理比较

旧的 CLI 驱动程序	通用驱动程序
旧的驱动程序从 DB2 产品获得错误消息，然后将整个错误消息传递给应用程序	通用驱动程序不会重新创建由旧的 CLI/JDBC 产品发出的、预先存在的 SQL 错误代码。通用驱动程序有它自己定义的错误代码，其范围为＋/－4200 和＋/－4299
	由通用驱动程序发出的未定义的错误代码被指定为－99 999
	如果收到来自一个 DB2 子系统(例如底层 DB2 客户机库的 DB2 服务器)的错误，那么 JCC 只是回显那条错误消息

- 事务管理

事务是一条或多条语句的集合，这些语句作为一个工作单元(UOW)一起执行。使用事务是为了确保一个 UOW 中的所有处理要么一起执行，要么都不执行。对于这些驱动程序，J2EE 指定了简单的事务管理。表 3.6 所示是旧的 CLI 驱动程序和通用驱动程序的事务管理比较。

表 3.6 事务管理

旧的 CLI 驱动程序	通用驱动程序
对于这类驱动程序，很早就已经提供了 XA 支持	从 V8.20 开始，Type 4 JDBC 通用驱动程序有了 XA 支持

3.2.2 诊断问题和分析跟踪

1. JDBC 跟踪的组成

在 DB2 中，无论何时收到任何类型的异常，接下来的一步就是找出那个错误的来源。在大多数情况下，要找出错误的起因，需要进行某种类型的跟踪，以便通过跟踪来揭示导致错误的调用序列。

来看看在 Java 中导致错误的调用序列，并研究在通常的 Java 应用程序中处理错误的机制。

在图 3.1 中可以看到，Java 运行时环境(JRE)包含用 Java 实现的错误处理机制。JRE 就像汽车中的引擎一样，使所有组件能够运行起来。

各组件可以由实际代码来表示，这些用 Java 编写的代码总是包含 try( )和 catch( )这两个代码块。每当实际代码碰到任何类型的错误，它就抛出一个异常，然后该异常就进入调用栈。调用栈将异常传递给 catch( )块，异常正是通过这种方式被返回给用户的。

为了允许 JDBC 程序抛出 SQLException，在技术实现上要确保程序能访问 com.ibm.db2.jcc.DB2Diagnosable 接口和 com.ibm.db2.jcc.DB2Sqlca 类。可以使用它们的完全限定引用，也可以导入它们：

```
import com.ibm.db2.jcc.DB2Diagnosable;
import com.ibm.db2.jcc.DB2Sqlca
```

2. SQLException 的组成

SQLException()类由以下几个部分组成：

```
SQLException(
Description of the error: null,string
SQL State: null,string
Error code: int value
Next SQLException: null or pointer
)
```

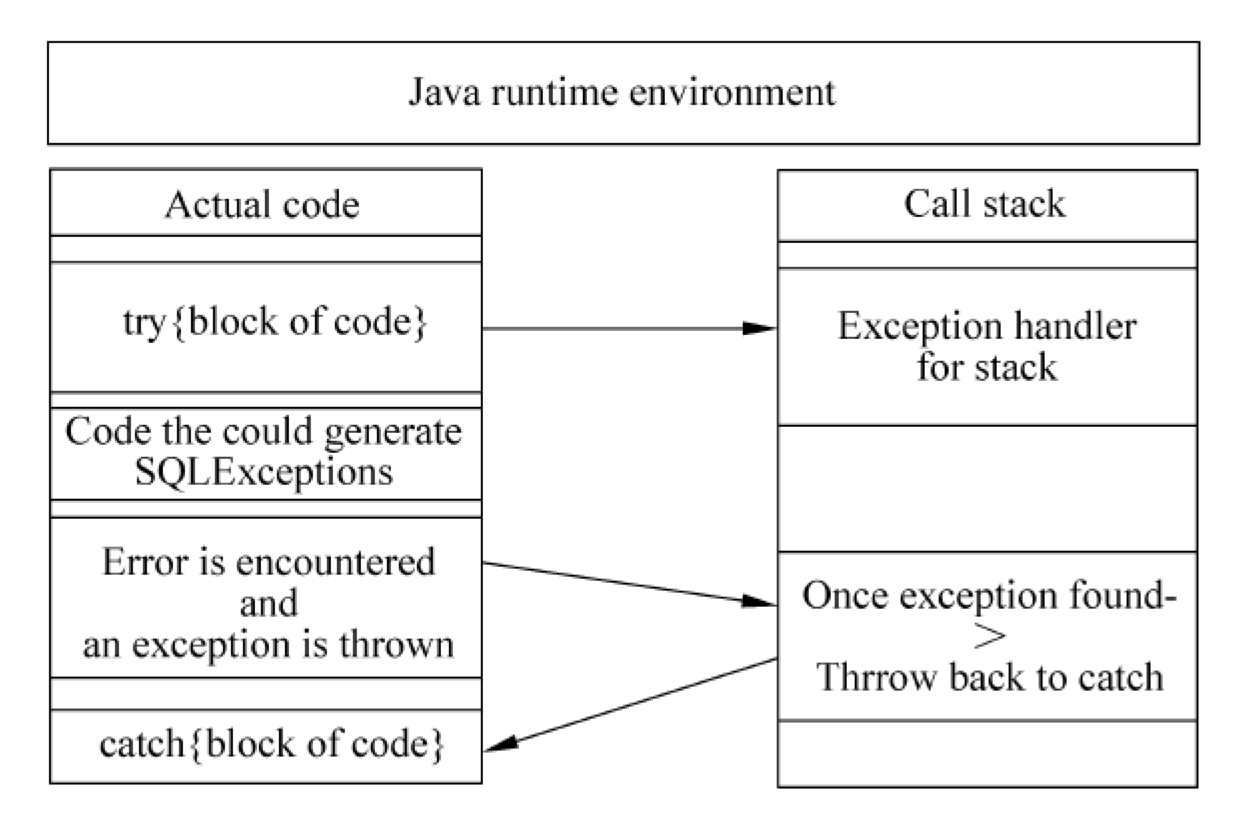


图 3.1 Java 运行时环境

3. 必要的存储过程

如果要使用通用 JDBC 驱动程序并连接到 OS/390,那么需要确保在主机上有一些必要的存储过程,这些存储过程将确保跟踪得以进行:

```
SQLCOLPRIVILEGES
SQLCOLUMNS
SQLFOREIGNKEYS
SQLGETTYPEINFO
SQLPRIMARYKEYS
SQLPROCEDURECOLS
SQLPROCEDURES
SQLSPECIALCOLUMNS
SQLSTATISTICS
SQLTABLEPRIVILEGES
SQLTABLES
SQLUDTS
SQLCAMESSAGE
```

这些存储过程是版本 6 的 PTF 附带的。需要 UQ72081 和 UQ72082。对于版本 7,PTF 号被定义为 UQ72083。如果需要关于如何安装这些 PTF 的具体信息,请参考 DB2 Information Center for z/OS,在那里可以获得详细的信息。

4. JCC 跟踪:概述

目前,JCC 驱动程序在跟踪和诊断问题方面的功能还不足以深入地诊断问题。目前的跟踪集还非常不稳定,主要是用于初步的分析。将来版本的 JCC 驱动程序将使跟踪功能更适合于问题诊断,并且更加面向问题。然而,JCC 跟踪中还是有几个关键的地方值得在后面加以讨论,它们有助于缩小问题的范围。

JCC 跟踪的实现有两种不同的方式，在接下来的两节中将详细讨论这两种方式。

如果用户曾经见过 DRDA 格式的 DB2 跟踪，那么 JCC 跟踪看上去会非常熟悉。从 DRDA 跟踪中获取缓冲区，然后将它们放入实际的 JCC 跟踪中。别忘了，JCC 使用 DRDA 来与服务器进行通信。

5. 如何进行 DB2 通用 JDBC 驱动程序跟踪

在跟踪一个 JCC 问题时，可以采取两种方法。根据环境的不同，可以使用以下两种方法之一：

- 把它作为一个独立的 JCC 应用程序来跟踪。
- 在 WebSphere 中，嵌入 JCC 跟踪点。

1）把 JCC 作为独立应用程序来跟踪

当把 JCC 组件作为独立应用程序来跟踪时，需要考虑与 DB2 通用 JDBC 驱动程序之间存在的连接类型。

（1）DataSource 接口

当为 JCC 连接使用数据源接口时，有两种方式来启用跟踪：

```
DB2DataSource > setTraceLevel > default TRACE_ALL
- javax.sql.DataSource.setLogWriter > TRACE_ALL only available
```

对于任何跟踪选项，除了 TRACE_ALL 属性外，还可以使用其他跟踪参数。根据所跟踪内容的不同，可以让 JCC 跟踪功能只跟踪以下属性：

```
com.ibm.db2.jcc.DB2BaseDataSource.TRACE_NONE
com.ibm.db2.jcc.DB2BaseDataSource.TRACE_CONNECTION_CALLS
com.ibm.db2.jcc.DB2BaseDataSource.TRACE_STATEMENT_CALLS
com.ibm.db2.jcc.DB2BaseDataSource.TRACE_RESULT_SET_CALLS
com.ibm.db2.jcc.DB2BaseDataSource.TRACE_DRIVER_CONFIGURATION
com.ibm.db2.jcc.DB2BaseDataSource.TRACE_CONNECTS
com.ibm.db2.jcc.DB2BaseDataSource.TRACE_DRDA_FLOWS
com.ibm.db2.jcc.DB2BaseDataSource.TRACE_RESULT_SET_META_DATA
com.ibm.db2.jcc.DB2BaseDataSource.TRACE_PARAMETER_META_DATA
com.ibm.db2.jcc.DB2BaseDataSource.TRACE_DIAGNOSTICS
com.ibm.db2.jcc.DB2BaseDataSource.TRACE_SQLJ
com.ibm.db2.jcc.DB2BaseDataSource.TRACE_XA_CALLS (仅适用于 Universal Type 2 Connectivity
for DB2 UDB for Linux、UNIX 和 Windows)
com.ibm.db2.jcc.DB2BaseDataSource.TRACE_ALL
```

如果想跟踪不止一个特定的 traceLevel 属性，那么可以使用位操作符(|)来分隔不同的属性。通常，如果不知道想跟踪哪个特定的组件，那么最好使用默认属性，即 TRACE_ALL。实际上，在大多数情况下，只需要知道这么多。但是如果需要更详细地规定跟踪某些 JDBC 通用驱动程序组件，那么可以通过位操作符来做到这一点。

另外提醒一点，如果想跟踪除了某个组件之外的所有组件，还可以使用另一个位操作符。这个位操作符就是(～)。

（2）DriverManager

进行跟踪的第二种方法是使用连接的 DriverManager 接口，可以通过以下两种方法之一来打开跟踪：

```
DriverManager.getConnection
```

设置 info 参数或 URL 参数中的 traceLevel 属性。

```
DriverManager.setLogWriter
```

当使用这种方法打开跟踪时，可以指定跟踪目标，然后打开跟踪。下面是关于如何做到这一点的一个好例子：

清单　使用 DriverManager.setLogWriter 的示例代码清单

```
//The traceLevel property is established through the URL syntax,
//and driver tracing is directed to file "/temp/driverLog.txt"
String databaseURL =
"jdbc:db2://sysmvs1.stl.ibm.com:5021" +
"/sample:traceFile = /temp/driverLog.txt; traceLevel = " +
"(com.ibm.db2.jcc.DB2BaseDataSource.TRACE_DRDA_FLOWS " +
"| com.ibm.db2.jcc.DB2BaseDataSource.TRACE_CONNECTS); ";
```

还有一种方法也可以进行 JCC 跟踪，这种方法不用修改应用程序。如果在客户机上创建一个名为 DB2JccConfiguration.properties 的文本文件，该文件中只有一行文本：

```
db2.jcc.override.traceFile = c:\jcc.trc
```

然后将该文件添加到 CLASSPATH 中，那么 JCC 跟踪将被自动启用。在不能更改任何源代码或 JCC 驱动程序属性的时候（例如，使用在内部使用 JCC 驱动程序的第三方产品），这种方法非常有用。

2）在 WebSphere 跟踪中嵌入 JCC 跟踪点

如果在 WebSphere 环境中碰到一个 DB2 通用 JDBC 问题，那么可以在 WebSphere 跟踪中嵌入 JCC 跟踪点。这样就可以很方便地知道 JCC 组件在 WebSphere 调用中所扮演的角色，从而可以清晰地了解应用程序中所发生的事情。

下面是在 WebSphere 跟踪中设置 JCC 跟踪点的步骤：

（1）在 WebSphere Application Server 中为 JDBC 设置跟踪属性。

```
进入 Resources > JDBC Provider > Data Sources > Additional Properties > Custom Properties.
```

需要设置的属性是：

```
traceLevel( -1 表示完全跟踪 TRACE_ALL)
```

（2）打开跟踪。

```
进入 Troubleshooting > Logs and Trace > pick the server > Diagnostic Trace > Trace
Specification: RRA = all = enabled: WAS.database = all = enabled
```

注意：这里可以指定两个跟踪字符串，之间用‘:’隔开，一个用于 WebSphere Application Server 资源适配器；另一个用于数据库(JDBC 驱动程序)。

通过使 traceFileName 属性空白，就足以自动在 WebSphere 跟踪中嵌入 JCC 跟踪点。可以动态地启用和禁用这种跟踪，在缩小问题范围的时候这样做会有所帮助。

3.2.3 JDBC 通用驱动程序错误代码

JCC 驱动程序只能发出少数几种 DB2 通用驱动程序错误代码。如果错误代码是通用驱动程序还没有定义的,那么它将回显一个−99 999 错误代码。表 3.7 中是 DB2 通用 JDBC 驱动程序当前可用错误代码的一个参考。

表 3.7 错误代码

代码	原　　因
4200	在 XA 环境中,一个处在全局事务中的应用程序发出一个无效的提交或回滚
4498	出现故障转移或故障恢复,事务失败
4499	导致连接断开的严重错误
−99 999	DB2 通用 JDBC 驱动程序发出没有错误代码的错误

目前由−99 999 这个通用错误代码定义的错误代码大约有 2000 条。下一阶段的 JCC 产品将用 SQLSTATE 和 SQLCODE 来定义这些错误代码。

在使用 JCC 驱动程序时,无论碰到何种类型的问题,为了做进一步的诊断,通常的做法是进行一个 JCC 跟踪。前面已经给出了进行 JCC 跟踪时所采取的步骤。现在来剖析一个 JCC 跟踪,看看如何通过分析跟踪和找出错误的来源来彻底把问题弄清楚。

现在,将一个跟踪分成几个部分,看看哪些部分在需要查看该诊断工具的各组件时会有用。在跟踪头中可以发现一些重要信息,这些信息对于理解环境非常有用。下面的编号对应图 3.2 中的数字。

(1) 所使用的 DB2 通用 JDBC 驱动程序版本

实际驱动程序版本是独立于修复包版本的。在 Java 应用程序开发支持页面上有一个详细的对照表,其中说明了每个 DB2 UDB 修复包所附带的 JCC 驱动程序的版本。

了解所使用的 JCC 驱动程序版本的另一种方法是在命令行中发出命令 db2jcc -version,该命令将显示当前使用的驱动程序的版本。

(2) JDK 级别

表明和该 JCC 驱动程序一起使用的 Java 开发工具箱的版本。尽量使之与所使用的相应修复包保持同步。

跟踪头中包含的其他重要信息有:

操作系统的级别

路径信息

获得最新版本的 DB2 通用 JDBC 驱动程序的最好方法是下载最近用于 DB2 UDB for Linux、UNIX 和 Windows 的修复包。

之所以要为 JDBC 驱动程序和相应的修复包使用不同的版本控制系统,是为了允许在所有 DB2 平台(包括 zSeries®、iSeries™等)上发布同一个驱动程序。这个驱动程序在所有 DB2 平台上是一致的。

现在来看看 JCC 跟踪的主体,并试着将一些关键的元素拼接起来。

(3) 跟踪标记

通过查看 JCC 跟踪中的标记,总可以确定所使用的通用驱动程序是 Type 4 风格的还

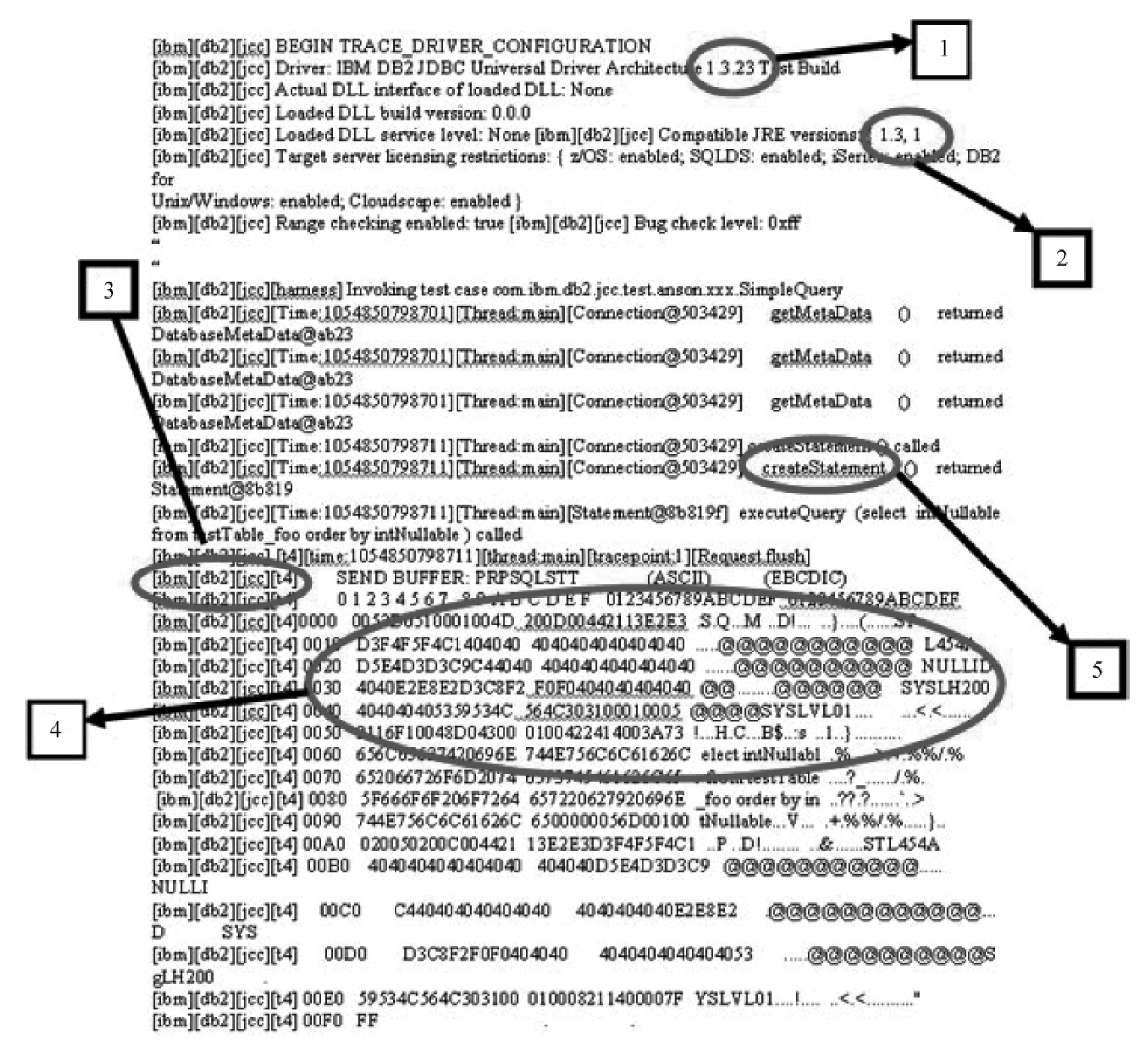


图 3.2 JCC 跟踪

是 Type 2 风格的：

[ibm][db2][jcc][t4] ＝表明所使用的是 type 4 版本的驱动程序。

[ibm][db2][jcc][t2] ＝表明所使用的是 type 2 版本的驱动程序。

(4) DRDA 缓冲区

由于 JCC 规范是建立在 DRDA 协议之上的，将 DRDA 缓冲区嵌入在 JCC 跟踪中。这些缓冲区包含诸如 PreparedStatement 对象或 ResultSet 对象之类的项。如果熟悉 DB2 跟踪中常见的 DRDA 缓冲区，那么 JCC 跟踪中的 DRDA 缓冲区的感观看上去会很熟悉。如果在查看 DRDA 信息时有些迷惑，那么可以抓住关键的一点，那就是要执行的 SQL 语句。这条语句应该就嵌入在缓冲区中，并和 DB2 通用驱动程序将它发送到服务器进行处理时是一样的。

(5) 使用的方法

如果知道导致发生问题的 Java 方法，或者想在跟踪中看看一个特定的方法是如何使用的，那么可以从 JCC 跟踪中找到它。

如果知道一个特定的语句或方法正在导致问题，那么总可以在 JCC 跟踪中搜索它，然后再对它进行上下搜索，从而发现任何值得怀疑的行为或错误消息，通过它们就可能找到能

指示出问题所在的线索。

如果对错误感到没把握，那么可以从 DB2 UDB Technical Support 站点开始着手。

现在来看一个有问题的跟踪的例子，该跟踪显示一个－4499 错误，这是由 DB2 通用 JDBC 驱动程序定义的一个错误代码。

通常，当碰到任何与通用 JDBC 驱动程序有关的问题时，都将以某种类型的异常的形式来报告这个问题。

从图 3.3 显示的跟踪中可以看到－4499 这个返回代码。异常中还显示了通信错误，可以看到，在这种特定情况下，这就是要返回给应用程序的内容。

一个很好的技巧就是上下搜索这个异常，掌握在实际应用程序中所发生的情况。通过搜索发现这是否是与驱动程序有关的缺陷，如果是，那么试着使用最新版本的 JCC 驱动程序，在最新版本中这个问题可能已经被修复了。

```
ibm][db2][jcc] BEGIN TRACE_DIAGNOSTICS
[ibm][db2][jcc][SQLException@68a936] java.sql.SQLException
[ibm][db2][jcc][SQLException@68a936] SQL state  = null
[ibm][db2][jcc][SQLException@68a936] Error code = -4499
[ibm][db2][jcc][SQLException@68a936] Message    = A communication error has been detected.
Communication protocol being used: Reply.fill(). Communication API being used: InputStream.read().
Location where the error was detected: Connection reset by peer. Communication function detecting the
error: *. Protocol specific error codes(s) TCP/IP SOCKETS
[ibm][db2][jcc][SQLException@68a936] Stack trace follows
com.ibm.db2.jcc.b.DisconnectException: A communication error has been detected. Communication
protocol being used: Reply.fill(). Communication API being used: InputStream.read(). Location where the
error was detected: Connection reset by peer. Communication function detecting the error: *. Protocol
specific error codes(s) TCP/IP SOCKETS
    at com.ibm.db2.jcc.a.a.a(a.java:428)
    at com.ibm.db2.jcc.a.ab.b(ab.java:175)
    at com.ibm.db2.jcc.a.ab.c(ab.java:222)
    at com.ibm.db2.jcc.a.ab.c(ab.java:337)
    at com.ibm.db2.jcc.a.ab.v(ab.java:1447)
    at com.ibm.db2.jcc.a.bb.b(bb.java:165)
    at com.ibm.db2.jcc.a.p.b(p.java:32)
    at com.ibm.db2.jcc.a.b.1b(b.java:2196)
    at com.ibm.db2.jcc.b.o.n(o.java:1161)
    at com.ibm.db2.jcc.b.o.h(o.java:1073)
    at com.ibm.db2.jcc.b.o.i(o.java:1082)
    at com.ibm.db2.jcc.b.rc.1(rc.java:558)
    at com.ibm.db2.jcc.b.rc.e(rc.java:465)
    at com.ibm.db2.jcc.b.rc.c(rc.java:350)
    at com.ibm.db2.jcc.b.rc.next(rc.java:277)
    at ifsvrComDBConnectItems.isGetSENDQUE_FTP(ifsvrComDBConnectItems.java:286)
    at ifsvrFtpDisk00.sqlExecuteCtl(ifsvrFtpDisk00.java:433)
    at ifsvrFtpDisk00.run(ifsvrFtpDisk00.java:215)
    at java.lang.Thread.run(Thread.java:479)
[ibm][db2][jcc] END TRACE_DIAGNOSTICS
```

图 3.3　跟踪例子



第4章

集成数据管理

现今企业所面临的难题是管理互联网和电子商务所造成的数据量和多样性的爆炸。原本存在于DBMS、内容管理系统、中间层高速缓存和数据仓库之间的界限正日益变得模糊，而迫切需要提供所有这些服务的统一视图的平台。这里提出了一种平台，它扩展了最新的联邦数据库体系结构，支持关系和XML作为主要数据模型，并紧密地联系了内容管理服务、工作流、消息传递、分析和其他企业应用程序服务。本章将对集成数据管理进行介绍。

4.1 集成数据管理简介

使用集成的模块化的数据管理环境，来实现一个跨整个数据管理生命周期设计、开发、部署、运营、优化和治理数据、数据库和数据驱动的应用程序。这个过程称为集成数据管理。通过关注整个生命周期和支持不同角色之间的协作，能够提高组织的生产力、增强敏捷性、提高效率，同时也改进服务质量、减少拥有成本，从而实现良好地治理各种数据、数据库和数据驱动的应用程序。

4.2 数据建模和设计

数据建模是一种用于定义和分析数据的要求和所需相应支持的信息系统的过程。因此，在数据建模的过程中，所涉及的专业数据建模工作，与企业的利益和用户的信息系统密切相关。

从需求到实际的数据库，有三种不同的类型。用于信息系统的数据模型作为一个概念数据模型，本质上是一组记录数据要求的最初的规范技术。数据首先用于讨论适合企业的最初要求，然后被转变为一个逻辑数据模型，该模型可以在数据库中的数据结构概念模型中实现。一个概念数据模型的实现可能需要多个逻辑数据模型。数据建模中的最后一步是确定从逻辑数据模型到物理数据模型的过程中，对数据、访问、性能和存储的具体要求。数据建模定义的不只是数据元素，也包括它们的结构和它们之间的关系。

4.2.1 数据仓库设计和数据建模

业务环境是在快速变化的，而业务数据的类型也是如此。一个成功的数据仓库解决方案的基础就是灵活的设计，这种设计可以适应不断变化的业务数据。数据仓库的架构和仓库数据的建模是仓库设计中的核心过程。

(1) 数据仓库的架构

当使用数据模型捕获业务需求时，用户就已经完成了数据仓库设计中的部分工作。然而，正式的数据仓库设计应该从数据仓库的架构开始。仓库架构是基于一些因素所做的关键决策，这些因素包括当前基础设施、业务环境、期望的管理和控制结构、实现工作的承诺和范围、企业采用的技术环境的功能以及可用的资源等。

(2) 架构选择

仓库架构将决定数据仓库和数据集市本身的位置，以及控制所驻留的位置，或者反之。例如，数据可以驻留在集中进行管理的中心位置中，数据也可以驻留在集中或独立管理的分布式的本地和(或)远程位置中。

有以下一些架构选择：

- 业务范围的数据仓库
- 独立的数据集市
- 互连的数据集市

这些架构选择也可以组合使用。例如，数据仓库架构可以在物理上分布或集中管理。

(3) 业务范围的数据仓库架构

业务范围的数据仓库就是支持整个或一大部分业务的数据仓库，该业务需要更加完全集成的数据仓库，跨部门和业务线(line of business)具有较高的数据访问和使用率，即基于整个业务需求设计和构造仓库。可以将之视作可跨整个企业使用的决策支持数据的公共存储库，或其中的一个大型子集。这里所使用的术语“业务范围(business-wide)”反映的是数据访问和使用的范围，而非物理结构。在整个企业中，业务范围的数据仓库在物理上可以是集中式的，也可以是分布式的。

(4) 独立的数据集市架构

独立的数据集市架构暗指单独的数据集市，这些数据集市是由特定的工作组、部门或业务线进行控制的，完全是为满足其需求而构建的。实际上，它们甚至与其他工作组、部门或业务线中的数据集市没有任何连通性，如图4.1所示。

(5) 互连的数据集市架构

互连的数据集市架构基本上是分布式的实现。虽然不同的数据集市是在特定的工作组、部门或生产线中实现的，但它们可以是集成、互连的，以提供更加全局的业务范围的数据视图。实际上，在最高的集成层次上，它们可以成为业务范围的数据仓库。这意味着一个部门中的终端用户可以访问和使用另一部门数据集市中的数据。

(6) 选择架构

如果客户的业务和数据源是相对集中的，那么业务范围的集中式数据仓库架构就是最明智的选择。这实际上对于中间市场的公司而言是很普遍的情况。否则，对于在地理上广泛分布的业务而言，互连的数据集市和业务范围的分布式数据仓库就是更加实用的选择。

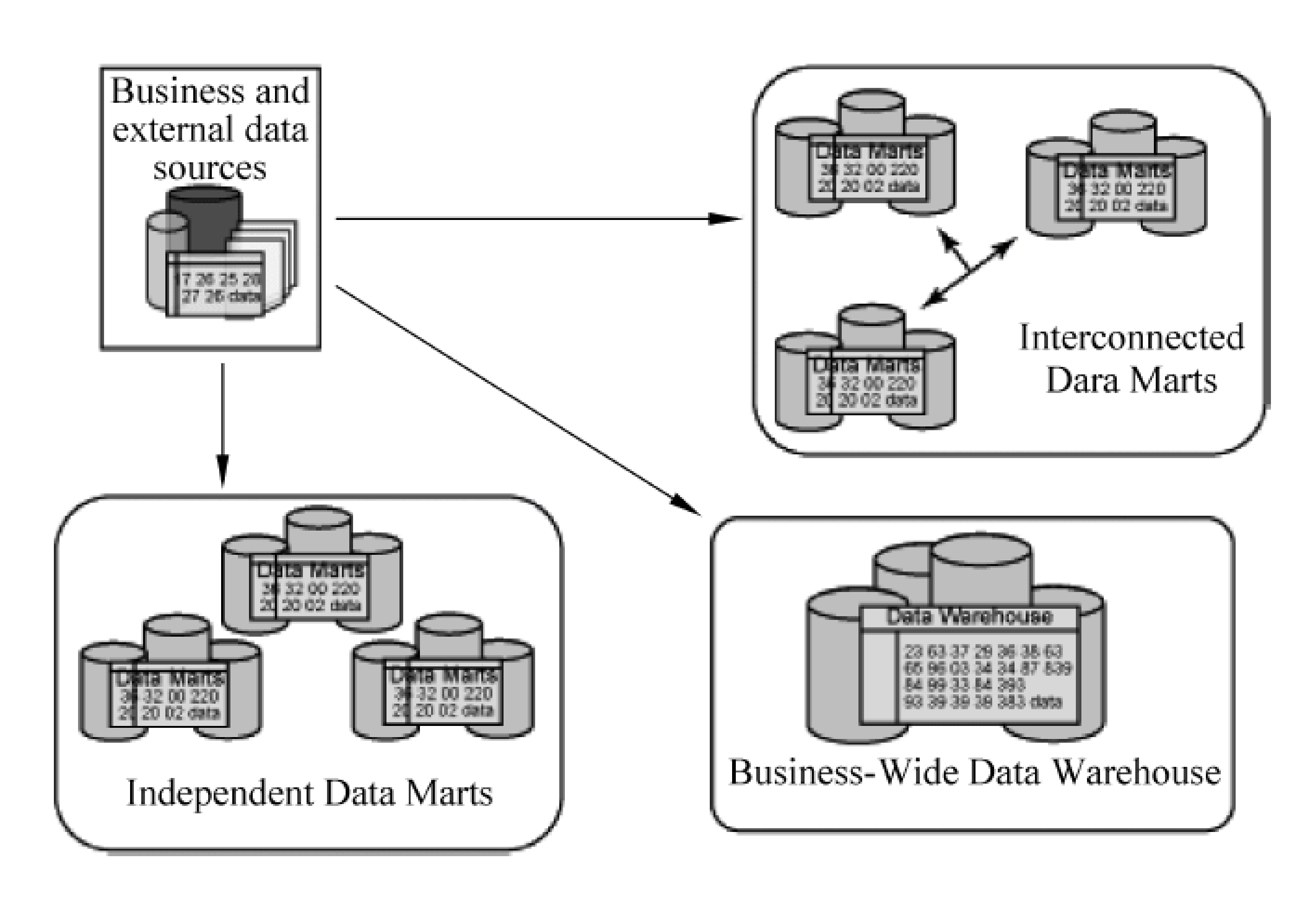


图 4.1　数据仓库架构选择

独立的数据集市架构不是一种好方法，因为它违背了数据仓库的关键概念：数据集成。

(7) 数据仓库的实现方法

实现方法的选择受这些因素的影响：当前的 IT 基础设施、可用的资源、所选择的架构、实现的范围、对于跨企业进行的更大业务范围的数据访问的需求、投资回报率(return-on-investment)需求以及实现的速度。实现方法的选择是数据仓库设计中的重要部分；该决策需要在数据仓库项目的早期阶段做出。

(8) 自顶向下的实现

自顶向下的方法就是在单个项目阶段中实现数据仓库。自顶向下的实现需要在项目开始时完成更多计划和设计工作。这就需要涉及参与数据仓库实现的每个工作组、部门或业务线中的人员。要使用的数据源、安全性、数据结构、数据质量、数据标准和整个数据模型的有关决策一般需要在真正实现开始之前就完成。

(9) 自底向上的实现

自底向上的实现包含数据仓库的计划和设计，无须等待安置好更大业务范围的数据仓库设计。这并不意味着不会开发更大业务范围的数据仓库设计；随着初始数据仓库实现的扩展，将逐渐增加对它的构建。现在，该方法得到了比自顶向下方法更广泛的认可，因为数据仓库的直接结果可以实现，并可以用作扩展更大业务范围实现的证明。

(10) 选择实现

每种实现方法都有利弊。在许多情况下，最好的方法可能是某两种的组合。该方法的关键之一就是确定业务范围的架构需要用于支持集成的计划和设计的程度，因为数据仓库是用自底向上的方法进行构建。

在使用自底向上或阶段性数据仓库项目模型来构建业务范围架构中的一系列数据集市时，用户可以一个接一个地集成不同业务主题领域中的数据集市，从而形成设计良好的业务数据仓库。这样的方法可以极好地适用于业务。每个数据集市都可以处理可识别的业务问题或主题领域，从而可以计算 ROI。构建团队可以测试并调整产品，而该方法也为构建团队提供了宝贵的学习曲线。

对于中间市场的公司，有一些额外的理由要采用自底向上的方法：

- 在中间市场的业务及其业务数据结构中，存在比企业业务数据中更多的易变性。
- 较小的公司通常存在有限的项目预算。
- 中间市场的公司需要快速解决方案以降低其业务难度。
- 该类项目所需要的人员必须具有对业务的广泛理解以及特定业务领域的详细知识。找到这样的人是很困难的，但即使可以，使用他们的时间来进行数据建模也比让他们尽普通业务职责更加困难。

(11) 数据仓库基础设施

做出具有关于高级数据仓库架构的一些决策之后，用户就可以开始考虑数据仓库应该具有什么组件了。

数据仓库应具有图 4.2 中所描述的商业智能基础设施中的所有组件。这里将关注数据仓库的构造，其中涉及了除信息分析之外的所有这些组件。

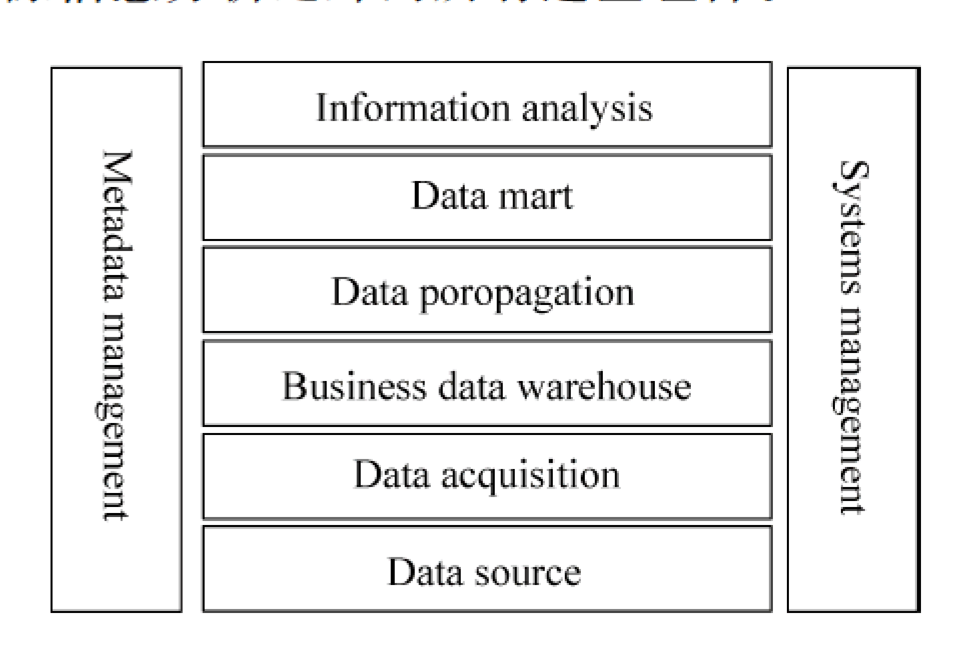


图 4.2 商业智能基础设施组件的高级视图

这些商业智能组件可以定义为：

- 数据源：当前可用的业务数据源和外部数据源以及将来可能存在的数据源的清晰定义。
- 数据获取：用于获得、清洗、转换和集成数据的 ETL(提取、转换和装入)过程。
- 业务数据仓库：仓库数据存储库的数据库。
- 数据传播：用于为数据集市聚集和增强数据的 ETL 过程。
- 数据集市：是友好的数据结构中的数据仓库的子集。
- 信息分析：本解决方案中未涉及。
- 元数据管理：业务需求、数据模型、ETL 过程设计、用户手册，等等。
- 系统管理：数据管理、数据仓库安全性、系统和数据库的备份和恢复，等等。

(12) 数据仓库的建模

数据只是所有业务活动、资源以及企业结果的记录。数据模型是对那些数据组织良好的抽象，因此数据模型成为理解和管理企业业务的最佳方法是极其自然的。数据模型起到了指导或计划数据仓库的实现的作用。在真正实现开始之前，联合每个业务领域的数据模型可以帮助确保其结果是有效的数据仓库，并且可以帮助减少实现的成本。

目标仓库数据的建模是将需求转换成图画以及支持表示那些需求的元数据的过程。出于易读性目的，这里将关于需求和建模的讨论相分离，但实际上这些步骤通常是重叠的。一旦在文档中记录一些初始需求，初始模型就开始成型。随着需求变得更加完整，模型也会如此。

最重要的是向终端用户提供良好集成并易于解释的数据仓库的逻辑模型。这些逻辑模型是数据仓库元数据的核心之一。为终端用户提供的简单性以及历史数据的集成和联合是建模方法应该帮助提供的关键原则。

(13) 仓库数据的建模与操作数据库的建模

在建模的过程中,请记住下列问题:

- 数据仓库应该是面向终端用户的。在数据库操作中,用户不直接与数据库进行交互。他们使用应用程序,这些应用程序具有预先定义的或固定的查询。数据仓库的数据库,特别是数据集市,与终端用户非常接近,它通常不具有固定的查询。因此,它必须更易于理解。
- 数据仓库应该是为数据分析而设计的。终端用户几乎直接处理数据,而且没有固定的工作流。终端用户对在仓库中记录数据不感兴趣,但他们需要从中获得信息。他们向仓库提出问题,通过所提取的信息测试并验证假设,重新构造事件链,分析那些事件以检测可能的模式或季节性的趋势,以及为将来做出推断和设计。
- 终端用户的需求可能是模糊或不完整的。这些不完整的需求需要灵活的建模过程和适合于进化开发的技术。灵活的进化软件开发的风险是不连贯和不一致的终端结果。在开发数据模型时,肯定需要注意这些问题。
- 数据仓库是集成的数据库集合,而非单个数据库。应将它构想为单个信息源,用于整个企业中所有的决策支持处理和所有的信息应用程序。数据仓库是一个“有机”物,如果在开始时还不够大,就还会趋于变大。
- 数据仓库包含属于不同信息主题领域的数据。这些主题领域可以是将数据仓库逻辑划分成几个不同(概念的,甚至是物理的)数据库的基础。数据仓库还可以包含不同类别的数据。
- 数据仓库通常包含历史数据,而不是日常操作数据的快照(snapshot)。必要的遗留数据库可能不可用,或者可能无法在足够细的层次上捕获,除非花费金钱并付出努力来改变遗留输入环境。因此,数据仓库启用项目通常涉及业务过程和源应用程序的重组(reengineering)。

(14) 两层数据仓库设计

两层的仓库建模方法最适合于自底向上的实现,如图 4.3 所示。

数据存储库层,或后端层包含了所有的人工模型组件和完整的模型结构,它们处理所有必要数据源中的集成业务数据。数据存储库层具有两个数据模块:数据登台(staging)和数据存储库。数据分级模块存储所有数据源的初始数据和临时数据,以用于 ETL 处理。数据存储库存储所有的集成业务数据,它是数据获取 ETL 过程的最终目标。

数据集市层包含所有的数据集市,这些数据集市是数据存储库模块的子集,以便特定的终端用户组在其数据分析活动中使用起来足够简单。

因为数据仓库和数据集市在许多情形中是可以交换的概念,所以有必要基于两层的仓库模型指定它们的定义。在该模型中,业务数据仓库是数据存储库数据库(模块)的集合,而数据集市则是数据集市层中的数据库。数据集市中的业务数据只能来自业务数据仓库。

两层数据仓库设计的好处包括:

- 灵活且易于维护:可以随时根据用户的报表需求修改数据集市的数据结构。但是,

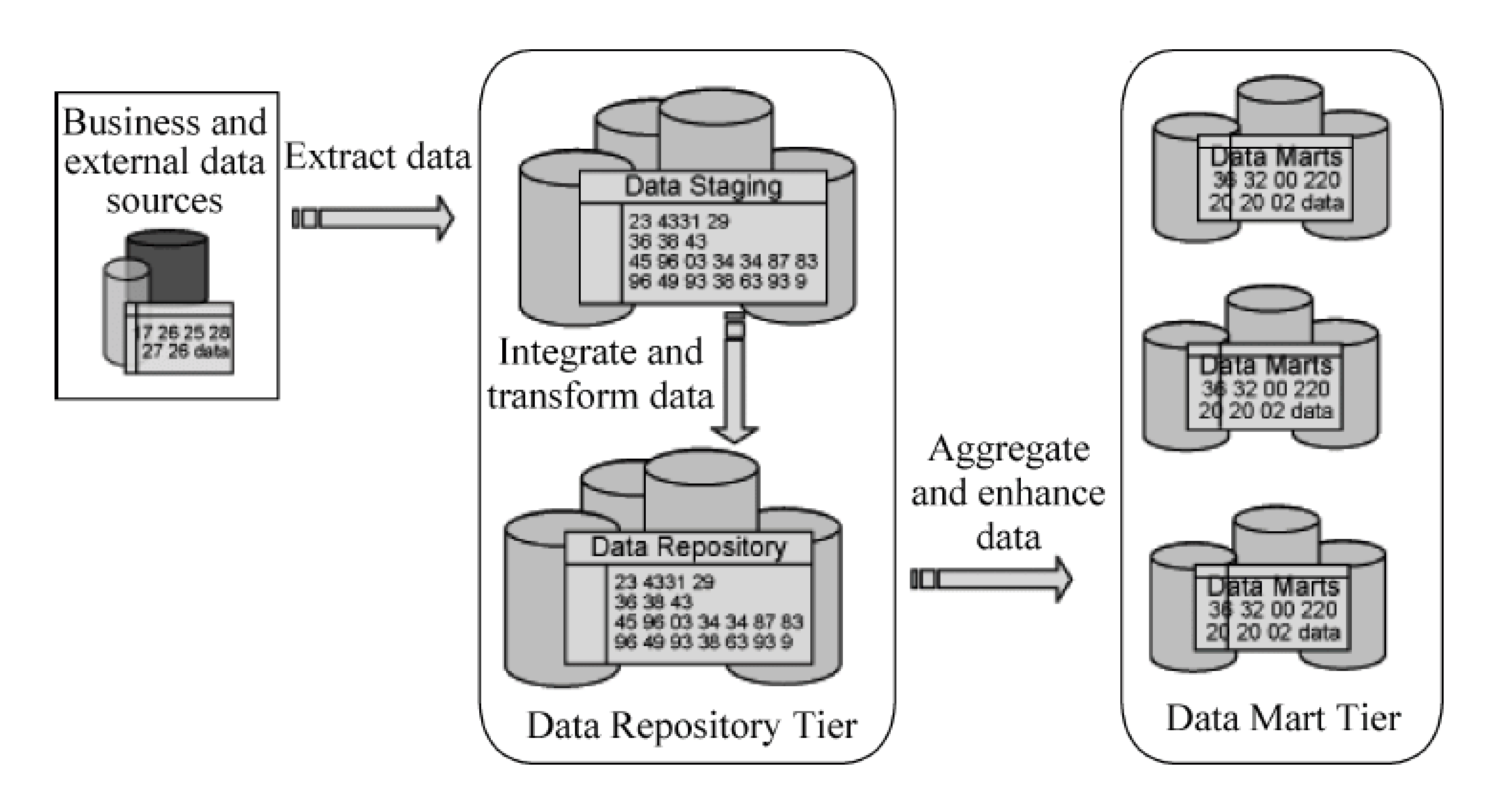


图 4.3　两层仓库建模

它不影响数据存储库中数据库的数据结构。

- 易于伸缩和集成：关系类型的业务数据存储库数据库比数据集市中非正规(denormalized)和汇总性(summarized)的数据库更易于伸缩和集成。
- 用户友好：将数据集市和数据存储库分离使得数据集市的设计可以完全由终端用户驱动，因为数据建模师不需要过多考虑数据集成和模式可伸缩性问题。
- 更好的安全设计：两层的方法将数据存储和数据访问管理相分离。终端用户只能访问授权给他们的数据集市，而非所有的数据仓库数据。
- 更好的数据管理设计：数据仓库是为存储集成的业务范围的历史数据而设计的。但是，数据仓库中的许多数据集市都不一定需要那么多的历史数据。两层设计是存储历史数据的好方法。

(15) 仓库数据建模层

数据建模有三层：概念、逻辑和物理。在数据仓库的设计中，数据建模的每一层都有自己的目的。

- 概念：高级数据模型是所有业务主题领域以及业务的公共数据元素的一致性定义，从高级的业务视图到通用的逻辑数据设计。用户可以从中派生出通用的范围，并获得对业务需求的理解。这个概念数据模型是当前和将来数据仓库开发阶段的基础。
- 逻辑：逻辑数据模型包含关于业务主题领域的更多详细信息。它捕获目标业务主题领域中的详细业务需求。逻辑数据模型是当前项目的物理数据建模的基础。

从这一阶段开始，该解决方案就适用自底向上的方法了，这意味着这个逻辑数据模型中仅仅将最重要和紧迫的业务主题领域锁定为目标。

逻辑数据模型的功能包括：

- ◆ 对于所有实体及其之间的关系的规范。
- ◆ 对于每个实体的属性的规范。
- ◆ 对于所有主键和外键的规范。
- ◆ 规范化(Normalization)和聚集。
- ◆ 对于多维数据结构的规范。

- 物理：物理数据建模应用物理约束，如空间、性能以及数据的物理分布。物理数据

模型是与数据库系统以及用户将使用的数据仓库工具紧密相关的。该阶段的目的是设计真正的物理实现。

将逻辑建模与物理建模清晰分离是特别重要的。良好的逻辑建模实践关注问题域的本质。逻辑建模解决"什么"之类的问题。物理建模为模型解决"如何"之类的问题,这表示给定的计算环境中实现的真实性。因为业务计算环境随时间变化,所以逻辑和物理数据建模的分离将帮助稳定一个阶段到另一阶段的逻辑模型。

一旦实现了数据仓库,且客户开始使用它,他们就常常会生成新的请求和需求。这将启动另一开发周期,继续数据仓库构建的迭代和进化过程。正如读者可以看到的,逻辑数据模型是数据仓库的活动部分,在数据仓库的整个生命周期中使用和维护。数据仓库的建模过程确实是无止境的,如图 4.4 所示。

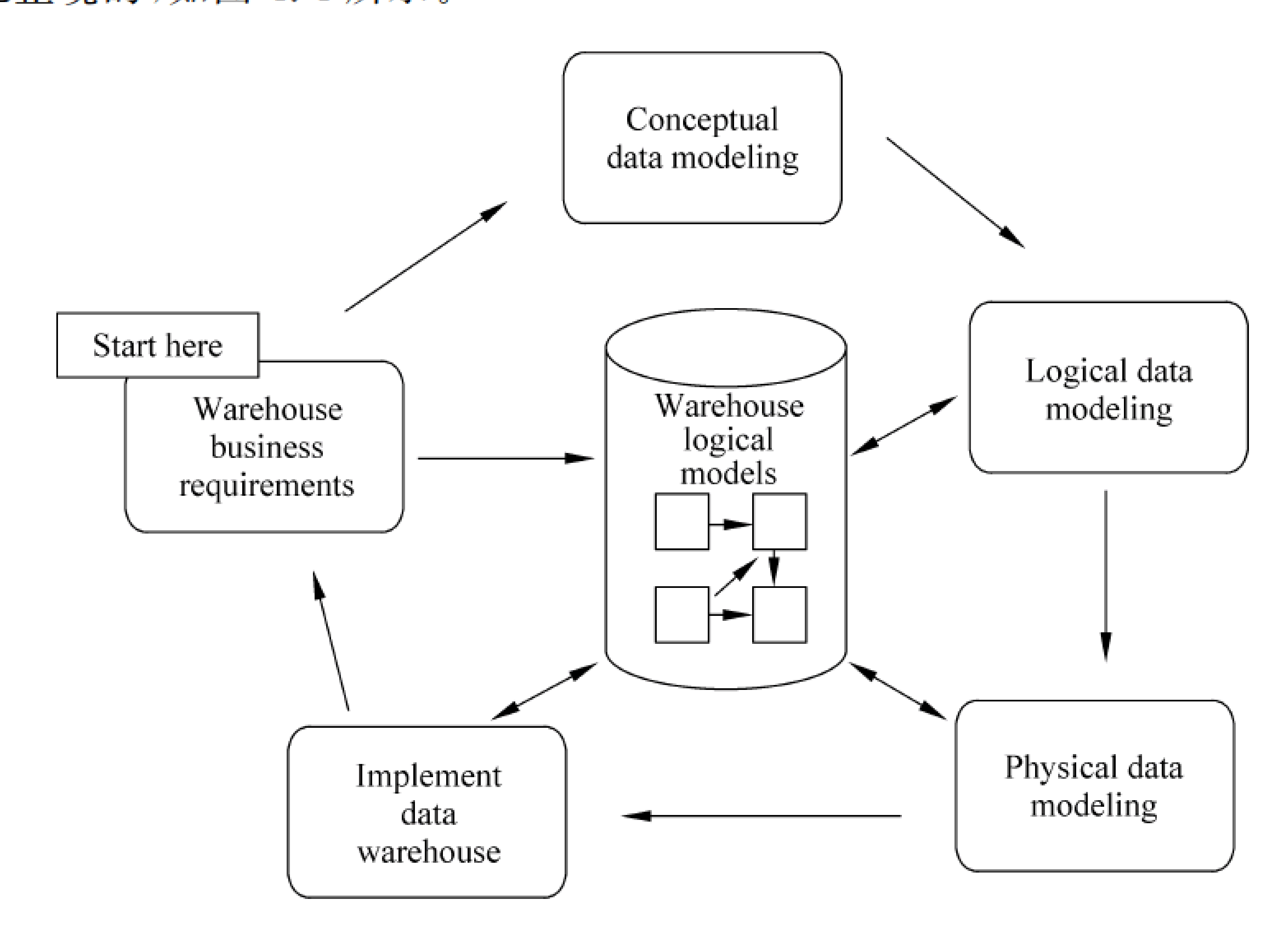


图 4.4　数据仓库逻辑数据模型的生命周期

(16) 存储库数据库建模

仓库数据存储库数据库存储来自所有业务数据源的所有清洁的(cleaned)集成业务数据,它是数据仓库中数据集成和转换的终点。虽然概念数据模型是根据业务需求设计的,但是数据存储库逻辑数据模型是基于业务需求和可用业务数据源的分析而设计的。它是用于验证现有业务数据是否支持数据仓库项目中业务需求的自然检查点。

存储库数据库基本上仍然是正规的(normalized)关系数据库。因为它们是源驱动的,所以源数据模型可以用于协助仓库数据存储库模型的完全开发过程。用户可能需要通过使用逆向工程(reverse engineering)技术构造源数据模型,从而从现有的源数据库中开发实体和关系(Entity and Relation,ER)模型。用户可能需要首先将几个这样的模型集成到一个全局模型中,以逻辑集成的方式表示源。

在数据转换过程中,清洗并清理了数据仓库中的数据。它至少应该与源操作系统中的数据一样好。在 ETL 过程中,即时参照完整性和检查约束极其有助于检测数据问题,但在最终的数据仓库表中实现它们也不会高效。

数据仓库的一个重要特点就是它包含历史数据。根据更新频率而言,有两种历史数据:

慢更新的和快更新的历史数据。对于慢更新的历史数据,更新是使用具有有效状态和时间框架(time-frame)数据的历史子表进行处理的。

事务和Web遍历数据是快更新数据的典型例子;它们通常具有较大的新旧数据量。存储快更新历史数据最重要的设计问题就是性能。例如,有从1999年开始的大量事务数据,但是正如业务需求所显示的,只有最近3年的事务数据才被频繁访问以制作报表。图4.5是一个高级的事务表的逻辑和物理分区设计。

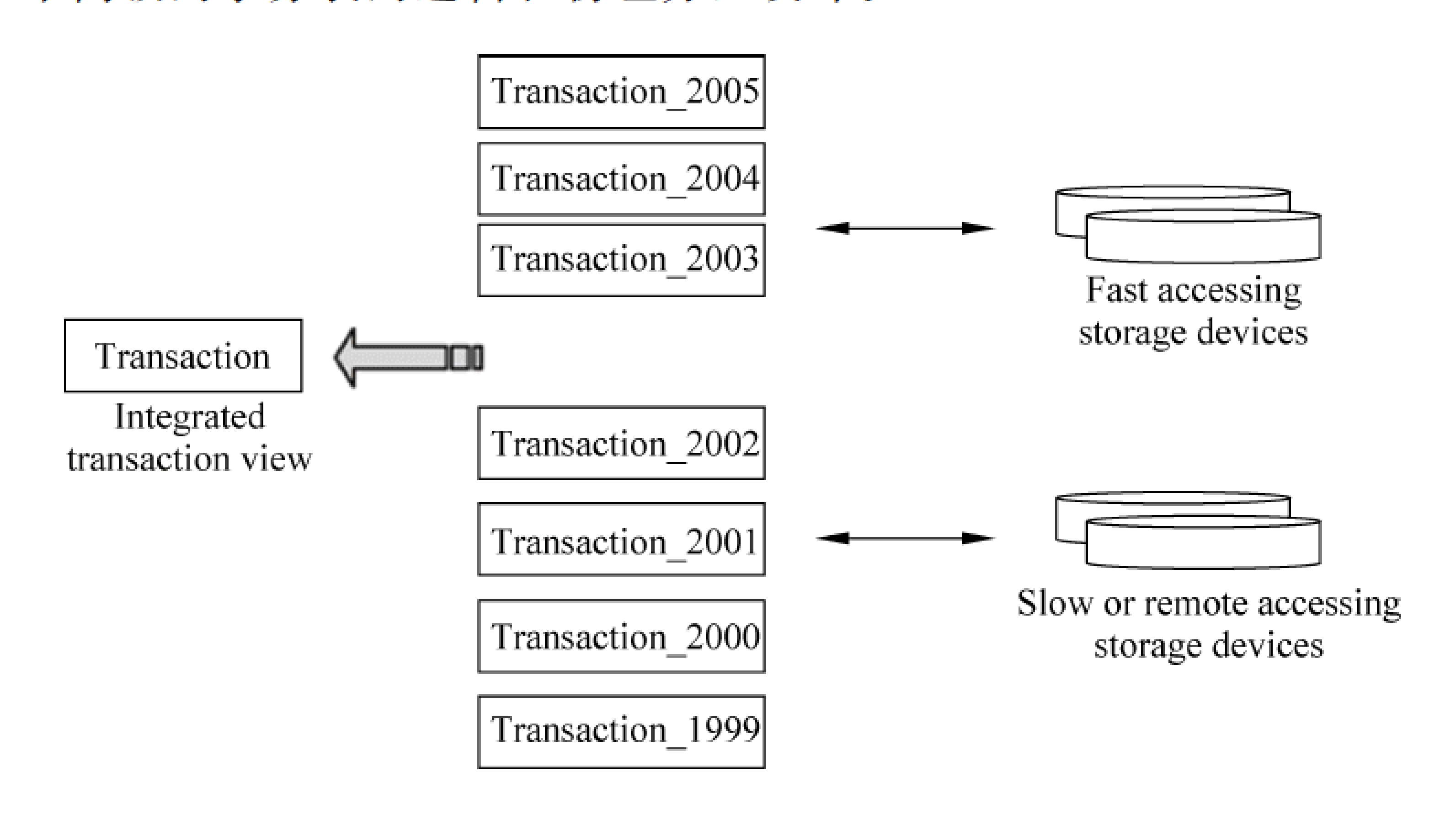


图4.5 事务表的逻辑和物理分区

(17) 数据集市的数据建模

因为仓库终端用户直接与数据集市进行交互,所以数据集市的建模是捕获终端用户业务需求的最有效工具之一。数据集市的建模过程取决于许多因素。下面描述了三个最重要的因素:

- 数据集市的建模是终端用户驱动的。终端用户必须参与数据集市的建模过程,因为他们显然是要使用该数据集市的人。因为应该期望终端用户完全不熟悉复杂的数据模型,所以应该将建模技术和建模过程作为整体进行组织,以便使复杂性对终端用户透明。
- 数据集市的建模是由业务需求驱动的。数据集市模型对于捕获业务需求十分有用,因为它们通常由终端用户直接使用,且易于理解。
- 数据集市的建模极大地受到了数据分析技术的影响。数据分析技术可以影响所选择的数据模型的类型及其内容。目前,有几种常用的数据分析技术:查询和报表制作、多维分析以及数据挖掘。

如果仅仅为提供查询和报表制作功能,那么带有正规(normalized)或非正规(denormalized)数据结构的ER模型就是最合适的。维度数据模型也可能是较好的选择,因为它是用户友好的,并具有更好的性能。如果其目标是执行多维数据分析,那么维度数据模型就是唯一的选择。然而,数据挖掘通常在可用的最低细节级(level of detail)工作得最好。因此,如果数据仓库是用于数据挖掘的,就应该在模型中包含较低细节级(level of detail)的数据。

因为这里没有包括ER建模,所以本小节将讨论维度数据建模,这是数据集市设计中最重要的数据建模方法。

(18) 星型和雪花型模型

有两种基本的数据模型是可以在维度建模中使用的：

- 星型模式(或模型)
- 雪花型模型

星型模式已经成为一个公共术语，用于表示维度模型。数据库设计师长期使用术语“星型模式”来描述维度模型，因为结果结构看上去像一颗星星。一些业务用户不习惯术语“模式”，所以他们接受听起来更加简单的术语“星型模型”。术语“星型模型”和“星型模式”是可以相互交换的。

星型模型是维度模型的基本结构。它通常有一个大型的中心表(称作事实表)和一组小型表(称作维度表)，维度表以放射模式围绕事实表，如图 4.6 所示。

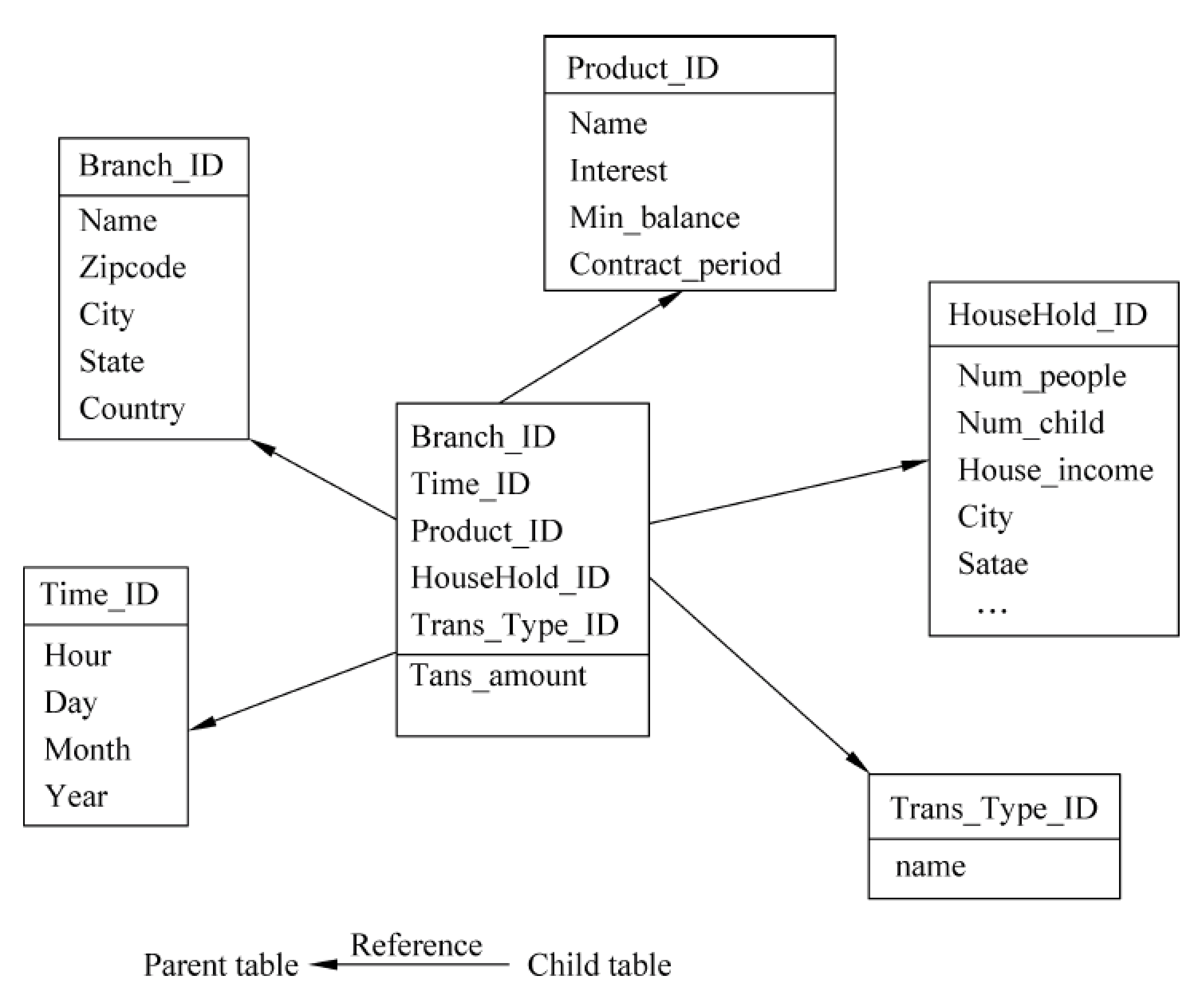


图 4.6 星型模式结构示例

传统的 ER 模型具有均匀且平衡的实体样式和实体之间的复杂关系，而星型模型却是完全不对称的。维度模型中的事实表与其他所有维度表之间存在单一连接。

维度建模通常在收集了业务需求之后首先指定事实和维度。初始的维度模型在外表上通常像星星一样，一个事实位于中心，一个级别上的多个维度则围绕在周围。雪花型模型是一个或多个维度的分解结果，它们自己有时具有层次结构。用户可以将一个维度表中成员之间多对一的关系定义为一个单独的维度表，从而形成层次结构。

分解的雪花型结构很好地展示了维度的层次结构。雪花型模型易于数据建模师的理解以及数据库设计师用于维度的分析。然而，雪花型的结构看起来更复杂，并且可能趋于使业务用户用起来感到不如更简单的星型模型舒服。开发人员也可以选择雪花型，因为它通常节省了数据存储空间。考虑一个银行应用程序，其中针对一个维度有一个极其大型的账户表。用户可以选择通过不存储极其频繁重复的文本字段，而是将其一次性放置在一个子维度表中，节省该大小的表中的大量空间。

虽然雪花型模型不节省空间，但与事实表相比通常是不重要的。大多数数据库设计师都不将节省空间作为选择建模技术的主要决策标准来考虑。

(19) 维度和测度

用户通常需要评估或分析企业业务的某些方面。所收集的需求必须表示该分析的两个关键元素：所分析的内容以及用于分析内容的评估标准。评估标准称作测度(事实的数字属性)，而所分析的内容称作维度(事实的描述属性)。一组维度及其相关的测度共同组成了所谓的事实。

(20) 维度

维度的基本结构就是层次结构。维度层次结构用于在比维度模型中测度所呈现的基本粒度更小的细节级(level of detail)来聚集业务测度，如销售总收入(Total Revenue of Sales)。本例中，操作称作上卷(roll-up)处理。上卷处理是对维度模型中的基本事实或测度执行的。

如果将测度上卷到更小的细节级(level of detail)，那么终端用户就显然可以执行逆操作(下钻)，该操作包含查看更详细的测度，或按照维度层次结构探索更低的细节级聚集测度。

维度建模最重要的活动之一就是捕获聚集路径或终端用户执行上卷和下钻所依照的聚集层次结构。该过程将产生维度模型，可以在稍后执行其他建模活动时扩展和修改该模型，如慢变化的时间维度的建模、维度中约束的处理以及跨维度的关系和约束的捕获。

维度建模与终端用户和业务过程紧密相关。为了使维度模型持续更长时间并适用于更多用户组，特别重要的是从概念的观点建模维度，寻找终端用户社区大致会感兴趣的基本的聚集路径和聚集级别。

用户通常可以向良好定义的事实添加测度，且根本不对模型造成很大影响。然而，对于维度，这就肯定不正确了，因为维度层次结构可能变得复杂。

如果要在广阔的环境中考虑问题域，将期望一个维度中具有多个不同的聚集路径。维度层次结构中的分割可以在不同的级别上出现。已经分割的层次结构可以稍后再次进行分割。该过程可能导致复杂的模式，也许太过于复杂，以致终端用户无法处理。

一个通常难以做出的重要决策就是一个聚集层是否真正是(结构化)层次结构的一个元素，或它是否仅仅是维度中一项的属性。例如，将产品包装单元、品牌或存储类型作为维度路径的显式元素(即，作为维度层次结构中潜在的实体类型)是否是明智或必要的呢？或者是否可以仅仅将它们考虑成产品的属性呢？

查找维度中的基本聚集路径通常意味着调查维度中的许多典型关系。

- 构造或结构化关系：信息分析员使用这些关系来探索产品及其组件之间的构造性关系。例如，通过与产品组件相关的成本和产品构造相关的成本，它们可以用于计算产品成本。
- 变化(Variation)关系：变化用于就产品模型、版本、实现、组件混合、调配等而言区分产品。

变化还可用于指定产品替换。信息分析员使用变化关系来组合相关产品和聚集相关测度，因为较低级类别的产品可能只存在于一段有限的时期内，或者因为它们频繁用于在某一过程中相互替换(例如，当“初始”产品缺货，而将某一版本的产品卖给客户时)。

- 分类关系：分类是将相似的产品分成组。分类关系显然是产品之间出现最频繁的关系，信息分析员用于上卷详细的测度。请注意，通常需要多个不同类型的分类。例如，可以根据面向销售、面向制造、面向储备或面向供应的特点来对产品进行分类。信息分析员将分类用于统计组中的聚集测度，统计组有总数、平均数、最小值和最大值等。

(21) 事实

事实表只包含用于引用维度表的 ID，以及用于测量所有维度成员变化或性能的测度。下一步就是将维度和测度组织成事实。这是以满足指定需求的方式将维度和测度进行分组的过程。

事实表的设计中要解决几个重要问题：

- 粒度(记录事实的细节级)。如果要有效地分析数据，就必须都在同一粒度级别上。一般说来，应该将数据放在最详细的粒度级别上。这是因为无法将数据修改到所决定设置的细节级上。但是总是可以上卷(汇总)数据以创建一个略粗的粒度级别的表。
- 相加性(要总结的测度的能力)。测度分成三个类别：全相加的(Fully Additive)、非相加的(Nonadditive)和半相加的(Semiadditive)。一个非相加测度的例子就是百分率。无法简单地将两个事实的百分率相加到一起，并产生有意义的结果。一个半相加测度的例子就是余额。虽然可以将两个账户的余额相加获得总的余额，但无法将同一账户在两个不同时间点的两个余额相加。因为余额只是跨一些维度进行相加的，所以称之为半相加测度。将可以跨所有维度相加的值视作全相加的。当考虑将在事实表上发生的可能汇总时，相加性就变得很重要。通常，全相加的测度是最理想的。当测度不是全相加时，应考虑将它们分成其原子元素。
- 键选择。多维数据建模中的键选择是一个难题。它包含性能和易于管理之间的权衡(Trade-Off)。键选择主要适用于维度。为维度所选择的键必须是事实的外键。维度键有两种选择：可以分配一个任意键，或者使用操作系统中的标识符。任意键通常只是一个序列号，当需要一个新键时，就分配下一个可用的号码。

为了使操作系统中的标识符唯一地表示维度，有时需要使用一个复合键。复合键就是由多个列组成的键。任意键是一列，通常比操作派生的键要小。因此，任意键通常可以更快地执行连接。

键选择中的最后一个因素就是它对事实表的影响。在创建事实时，必须将每个维度的键分配给它。如果维度将带有时间戳的操作派生的键用于历史数据，那么在创建事实时，就没有附加工作。连接将自动发生。对于任意键或任意历史标识符，在创建事实时，就必须将一个键分配给事实。

分配键的方式有两种。一种就是维护操作和数据仓库的键的转换表。另一种就是存储操作键，并且在必要时，存储时间戳作为维度上的属性数据。

那么，选择就在任意键的更好性能和操作键的更易维护之间进行。性能提高多少和维护增加多少的问题就必须在自己的组织中进行评估。

无论做出什么选择，都必须在元数据中用文档记录生成它们的过程。该信息对于管理和维护数据仓库的技术人员来说是必要的。如果所使用的工具没有隐藏连接处理，那么用

户可能也需要理解这一点。

既然理解了维度和事实表的处理，那么就来看一看真实世界的例子，以探索如何从业务需求中识别维度和测度。该例子只是对于一系列业务问题的基本分析。这些业务问题被定义为示例需求：

① 按照银行支行进行统计，本月客户的平均余额和交易数是多少？

② 按照支行、产品和地区汇总，要付给每位客户的年净利润和利息是多少？

③ 百分之几的客户是盈利的？将他们按照支行、地区和年分类。

④ 本年一位客户的总交易量是多少？

⑤ 按照地区，最盈利的前 5 位产品是什么？

⑥ 在最近 5 年里，最盈利的前 5 个支行是哪几个？

⑦ 最盈利的客户的人口和地理特点是什么？

通过分析这些问题，就定义出了需要满足需求的维度和测度（见表 4.1）。

表 4.1　维度和测度表

维度和测度	Q1	Q2	Q3	Q4	Q5	Q6	Q7
维度	?	?	?	?	?	?	?
支行	? X	? X	? X	?	?	? X	? X
地区	?	? X	? X	?	? X	?	? X
客户	? X	? X	? X	? X	?	?	? X
产品	?	? X	? X	?	? X	?	?
时间	? X	? X	? X	? X	?	? X	?
?	?	?	?	?	?	?	?
测度	?	?	?	?	?	?	?
余额	? X	?	?	?	?	?	?
交易量	?	?	?	? X	?	?	?
交易数	? X	?	?	?	?	?	?
净利润	?	? X	? X	?	? X	? X	? X
所付利息	?	? X	?	?	?	?	?

此时，要检查维度以确保：

- 具有用于回答问题的数据。
- 在最细的级别定义了所有的测度。

使用这些简化的分析问题来决定最终的星型模型中包含什么以及排除什么：

- 余额和交易数基于交易量的聚集，所以它们是派生的测度。
- 所付利息的计算是将账户利率乘以余额。这是基于账户和基于月份的计算。因为利率是 Account 表的一个属性，所以需要添加 Account 维度表。正如可以看到的，所付利息也是派生的测度。
- 假定净利润基于（投资收益）－（所付利息）的计算。因为投资收益是银行级的测度，而所付利息是派生的测度，所以净利润也是派生的测度。

从以上分析得出的结论有：

- 交易量是唯一需要的测度。
- 需要账户维度来产生利率和投资收益信息。

(22) 模型元数据

在传统的开发周期中,完成一个模型之后,只有需要进行修改或其他项目需要数据时,才使用它。但是在该仓库中,要连续使用模型。仓库的用户不断访问该模型以决定他们需要用于分析组织的数据。仓库中数据结构的修改速度比操作性数据结构要快得多。因此,仓库的技术用户(管理员、建模师、设计师等等)也将定期使用创建的模型。

这就是元数据所承担的任务。远远不只是一幅漂亮的图画,该模型必须是所存储数据的完整表示,否则,它对于任何人都毫无用处。

为了正确理解模型,以及可以确认它满足了需求,用户必须访问按照容易理解的业务术语描述仓库的元数据。因此,应在此时记录非技术的元数据。在设计阶段,将向它添加技术元数据。

在仓库级上,应提供仓库中可用东西的列表。该列表包含可用的模型、维度、事实和测度,因为当用户开始分析数据时,这些都将用作初始的进入点。

对于每个模型,提供名称、定义和目的。名称仅仅给用户提供一些搜索时关注的东西。它通常与事实相同。定义指定建模的内容,而目的描述模型用于什么。模型的元数据也应包含与之相关的维度、事实和测度列表,以及联系人员的姓名,以便用户在有关于模型的问题时,能够获得附加的信息。

(23) 模型验证

在投入大量时间和精力实现仓库数据库之前,与用户一起验证模型是一个很好的想法,特别是数据集市模型。进行该检查的目的是双重的。首先,它确认模型可以真正满足用户的需求。其次,检查应确认用户可以理解该模型。请记住,一旦构建了仓库,用户就会定期依靠模型访问仓库中的数据。无论模型多么好地满足了用户的需求,如果用户无法理解模型从而妨碍访问数据,那么这样的仓库都是失败的。

此时,验证是在高级别上完成的。与用户一起检查该模型以确认它是可以理解的。与用户一起,通过解决需求中指定的部分问题来测试模型。

模型不必满足用户的所有需求,这一点是好的。但这不意味着应该停止并返回到开始。期望模型的第一次切入满足约50%的需求。接受模型的这50%(或者无论验证了多少),并开始进行物理设计。应将剩余的送回给需求的收集阶段。要么需要更好地理解需求,要么通常需要修改并重新定义它们。这常常会导致对已创建的模型进行添加和修改。同时,模型的有效部分将通过设计阶段,并开始向用户提供便利。

开发的重复和部分完整模型的继续创建是提供快速开发数据仓库能力的关键元素。

在需求的验证过程中将:

- 对照给定的用户需求,检查初始维度模型的连贯性和完整性以及有效性。与终端用户一起分析初始模型。这将允许需求分析员执行更多调查,并在将其传递给需求建模阶段之前调整这些初始模型(尝试按照模型中所表达的内容修改需求)。
- 指定候选的数据源。建立必要和可用数据源的库存。
- 将初始维度模型和可能配有的信息化终端用户需求映射到指定的数据源。这通常是一项冗长乏味的工作。源数据映射必须调查下列映射问题:

① 哪些源数据项可用，哪些不可用？对于不可用的那些源数据项，是否应扩展源应用程序？是否可以使用外部数据源找到它们？或者是否应向终端用户通知它们的不可用性，并且因此，是否应减小维度模型的范围？

② 数据源中是否有其他可用但没有请求的有趣数据项呢？指定可用但没有请求的数据项可能暴露信息分析活动的其他有趣方面，并且可能因此极大地影响所构造的维度模型的内容和结构。

③ 验证数据集市设计没有违背任何业务安全性设置。因为数据集市是为特定的终端用户组设计的，所以确认它只包含该组必要的信息是很好的想法。

- 执行模型的初始大小调整。如果完全可能，初始的大小调整还应调查与填充数据仓库相关的容量和性能方面的情况。

需求验证的结果将帮助评估数据仓库开发项目的范围和复杂性，并（重新）评估业务的调整，包括技术、金融和资源的评估。需求验证必须与终端用户协同执行，以暴露和改正不完整或不正确初始模型的所有问题。需求验证可能涉及构建维度模型的原型。

需求验证过程将确认或重新建立终端用户需求和期望。作为需求验证的结果，还可能指定和评估源数据重组（Reengineering）建议。在需求验证的最后，将获得用于数据仓库建模项目的（新的）“签字”。仓库数据模型评估过程如图 4.7 所示。

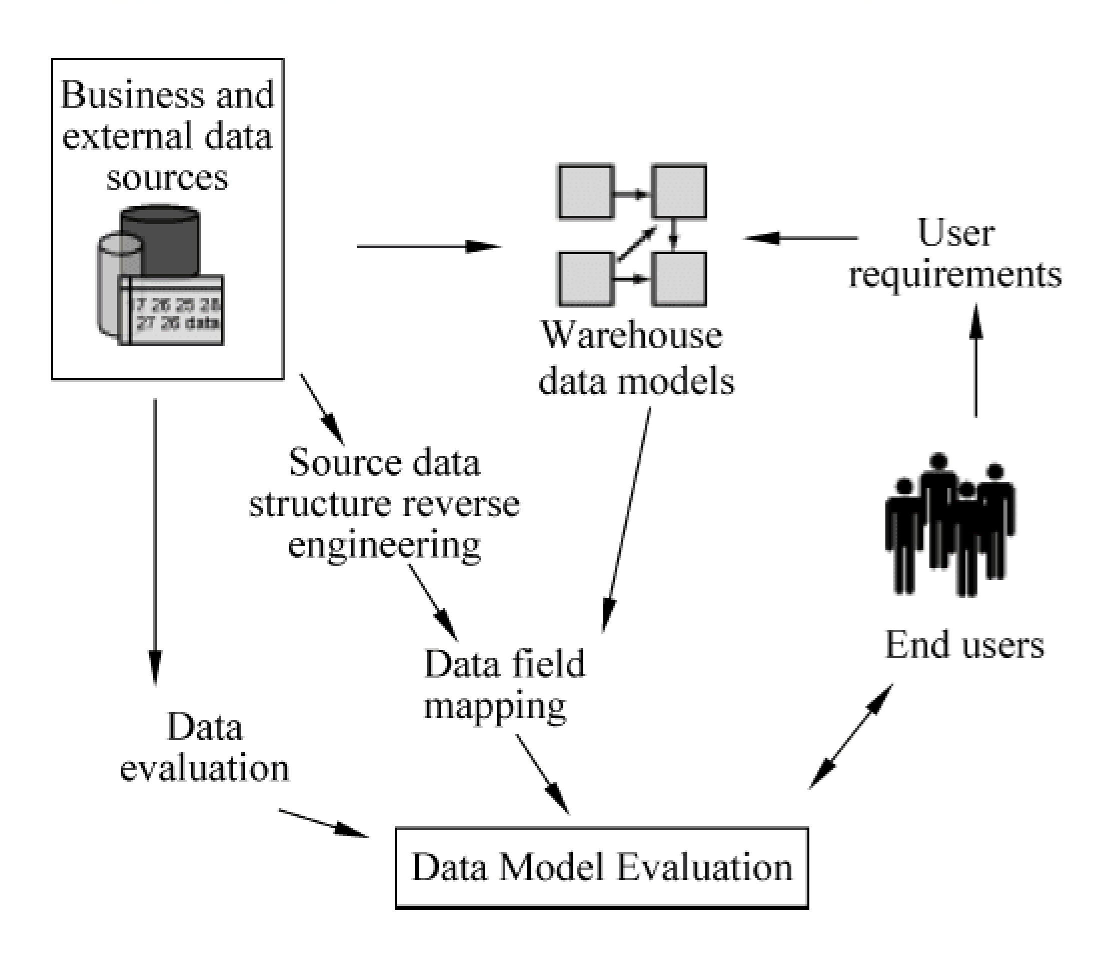


图 4.7　仓库数据模型评估过程

4.2.2　使用 IBM® InfoSphere® Data Architect 对 DB2 创建的全局临时表建模

（1）问题描述

如果某人是一家公司 IT 部门员工。公司花钱为其客户执行项目。该人的团队需要开发一个项目成本控制应用程序，该应用程序可生成项目执行报告，以便帮助管理部门制定决策。以下要求是应用程序必须考虑的。

- 企业可为同一个客户执行多个项目。
- 按照商定的条款，在完成项目后，客户需要支付一定的费用。
- 项目应有一个开始日期和一个结束日期。

- 必须将企业员工分配到具体的项目。员工在任一时间点只能从事一个项目工作。
- 企业员工需要按月支付薪酬。
- 项目费用分为以下几类：
  - 薪金
  - 原材料
  - 制造
  - 包装
  - 运输

管理人员需要使用以下报告进行分析和制定决策。

- 已完成项目的单独项目成本报表。
  - 项目 ID 和项目名称
  - 完成项目后的费用由客户支付
  - 各种项目成本费用
  - 项目执行总费用
  - 利润率—((价格—总产成本)/总成本)×100
- 已完成项目的财年整合报表。
  - 财务年度和执行的项目数
  - 项目收入
  - 项目执行总费用
  - 利润率—((项目收入—项目总成本)/项目总成本)×100
  - 目前正在运行的、成本已超出报价的项目
  - 项目 ID 和项目名称
  - 价格
  - 目前已发生的总成本
  - 成本超支
  - 客户 ID 和客户名称
- 为客户执行的完成项目利润率报表。
  - 客户 ID 和客户名称
  - 项目号
  - 执行项目的总收入
  - 执行项目已发生的总成本
  - 利润率

(2) 模型设计

可以创建一个简化版的逻辑数据模型 ProjectCostModel.ldm，它包含建模问题描述中列出的需求的字符实体、属性和关系，如图 4.8 所示。

具体项目 PROJECT 实体模型信息如下。

PROJECT_ID：项目的唯一标识符代码。

PROJECT_NAME：项目名称。

PROJECT_DESCRIPTION：项目描述。

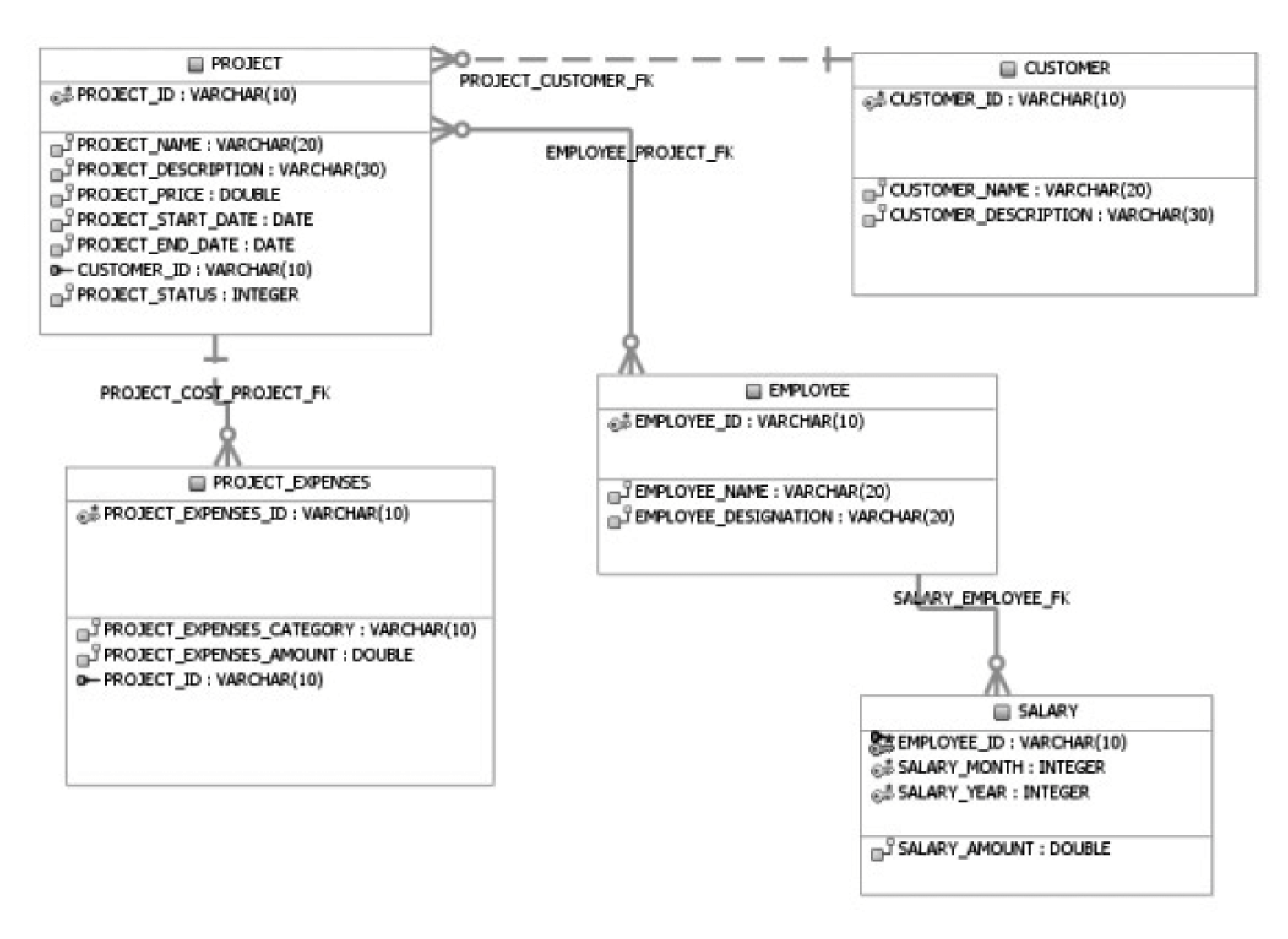


图 4.8　简化的项目成本控制系统逻辑数据模型

PROJECT_PRICE：项目价格。

PROJECT_START_DATE：项目开始日期。

PROJECT_END_DATE：项目结束日期。

PROJECT_STATUS：表明项目正在运行或已完成。

CUSTOMER_ID：客户的唯一标识符代码。

具体客户的 CUSTOMER 实体模型信息如下。

CUSTOMER_ID：客户的唯一标识符代码。

CUSTOMER_NAME：客户名称。

CUSTOMER_DESCRIPTION：客户描述。

特定员工的 EMPLOYEE 实体模型如下。

EMPLOYEE_ID：员工的唯一标识符代码。

EMPLOYEE_NAME：员工名。

EMPLOYEE_DESIGNATION：员工编号。

薪金水平 SALARY 实体模型信息如下。

EMPLOYEE_ID：员工的唯一标识符代码。

SALARY_MONTH：薪金支付月份。

SALARY_YEAR：薪金支付年份。

SALARY_AMOUNT：薪金数额。

具体项目非薪金费用 PROJECT_EXPENSES 实体模型信息如下。

PROJECT_EXPENSES_ID：项目费用唯一标识符代码。

PROJECT_EXPENSES_CATEGORY：薪金之外的其他费用种类。

PROJECT_EXPENSES_AMOUNT：费用总额。

PROJECT_ID：项目的唯一标识符。

(3) 将逻辑数据模型转化成物理数据模型

逻辑数据模型设计已得到业务用户的批准。要建模物理存储，必须将逻辑数据模型转化成所使用的 DBMS 物理数据模型。执行以下步骤进行模型转化。

① 右击选择【逻辑模型】，然后从上下文菜单中选择 Transform to Physical Data Model，如图 4.9 所示。

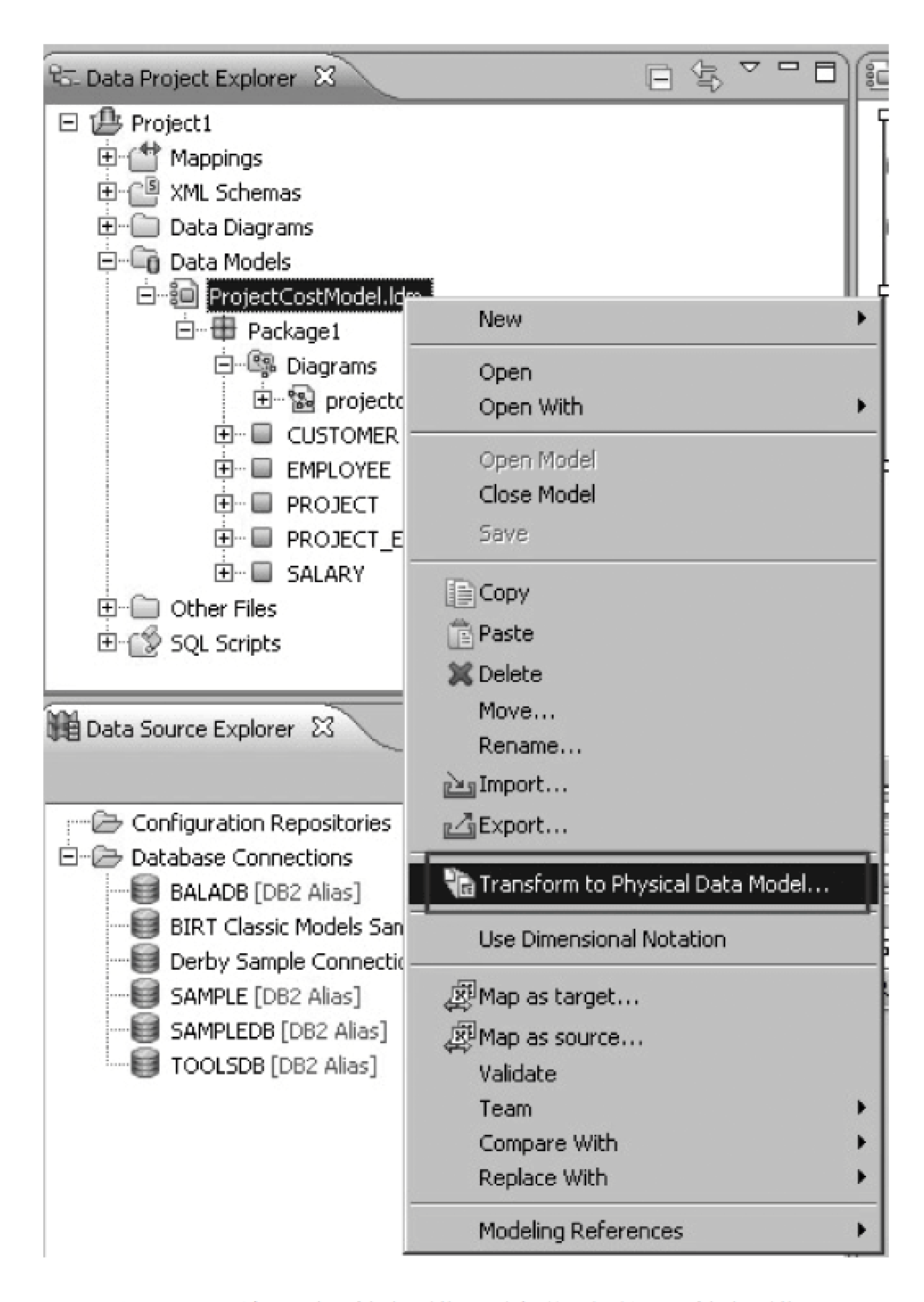


图 4.9　将逻辑数据模型转化成物理数据模型

② 从 Target Physical Data Model 页面中选择 Create new model 选项。

③ 从 Physical Data Model File 页面中指定文件名，然后选择所使用的数据库详细信息。在图 4.10 中，DB2 for z/OS V10(新功能模式)被选中。

④ 在 Options 页面中，在 Schema name 字段中指定 COST_MODEL_SCHEMA，如图 4.11 所示。其他选项保留默认值。

⑤ 在 Output 页面中查看转化状态，然后选择 Finish 生成物理数据模型。物理数据模型 ProjectCostModel.dbm 是从逻辑数据模型中创建的。

(4) 查看生成的物理数据模型

查看 Data Project Explorer 中的物理数据模型，如图 4.12 所示。

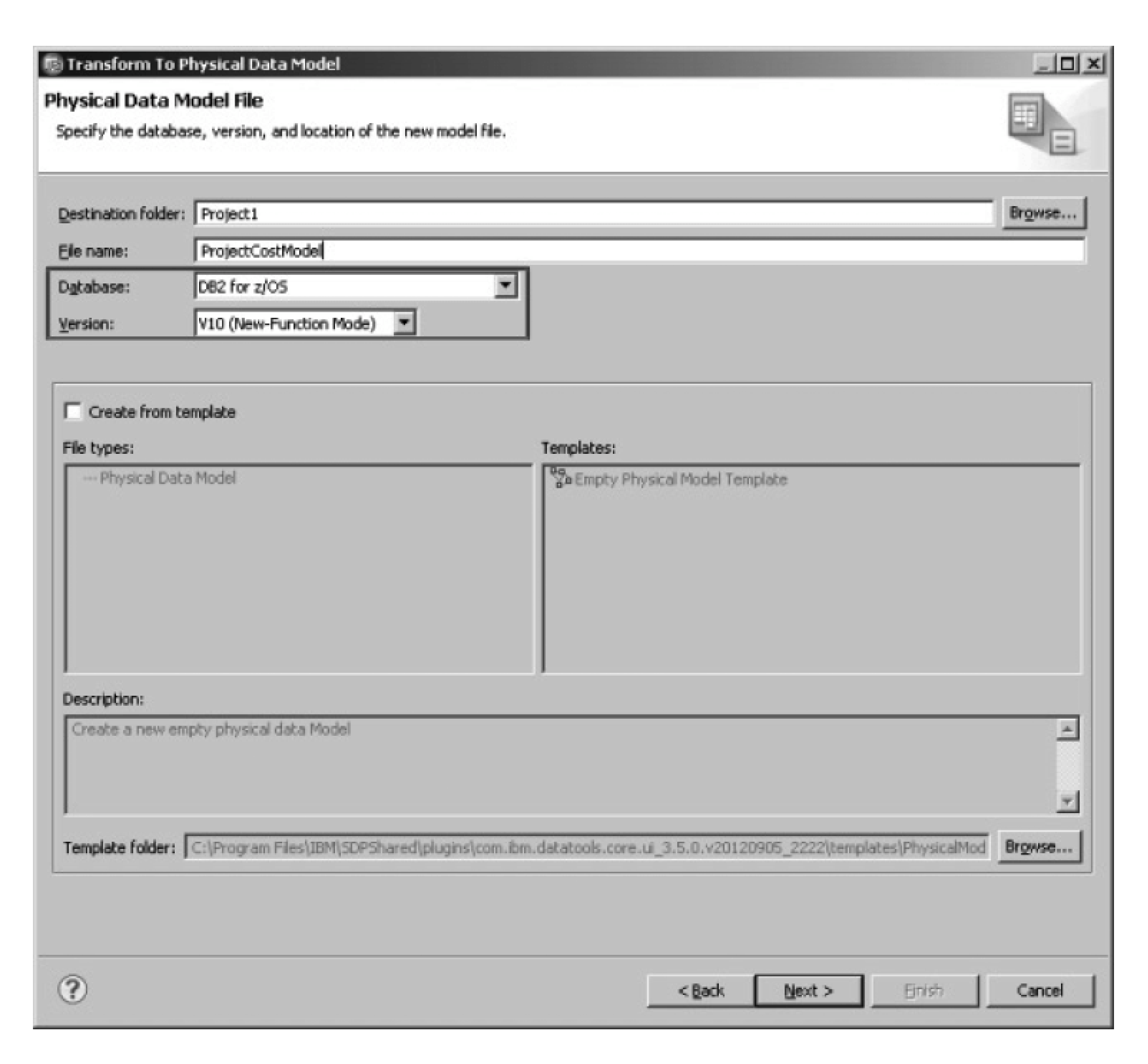


图 4.10　指定转换的数据库、版本和位置

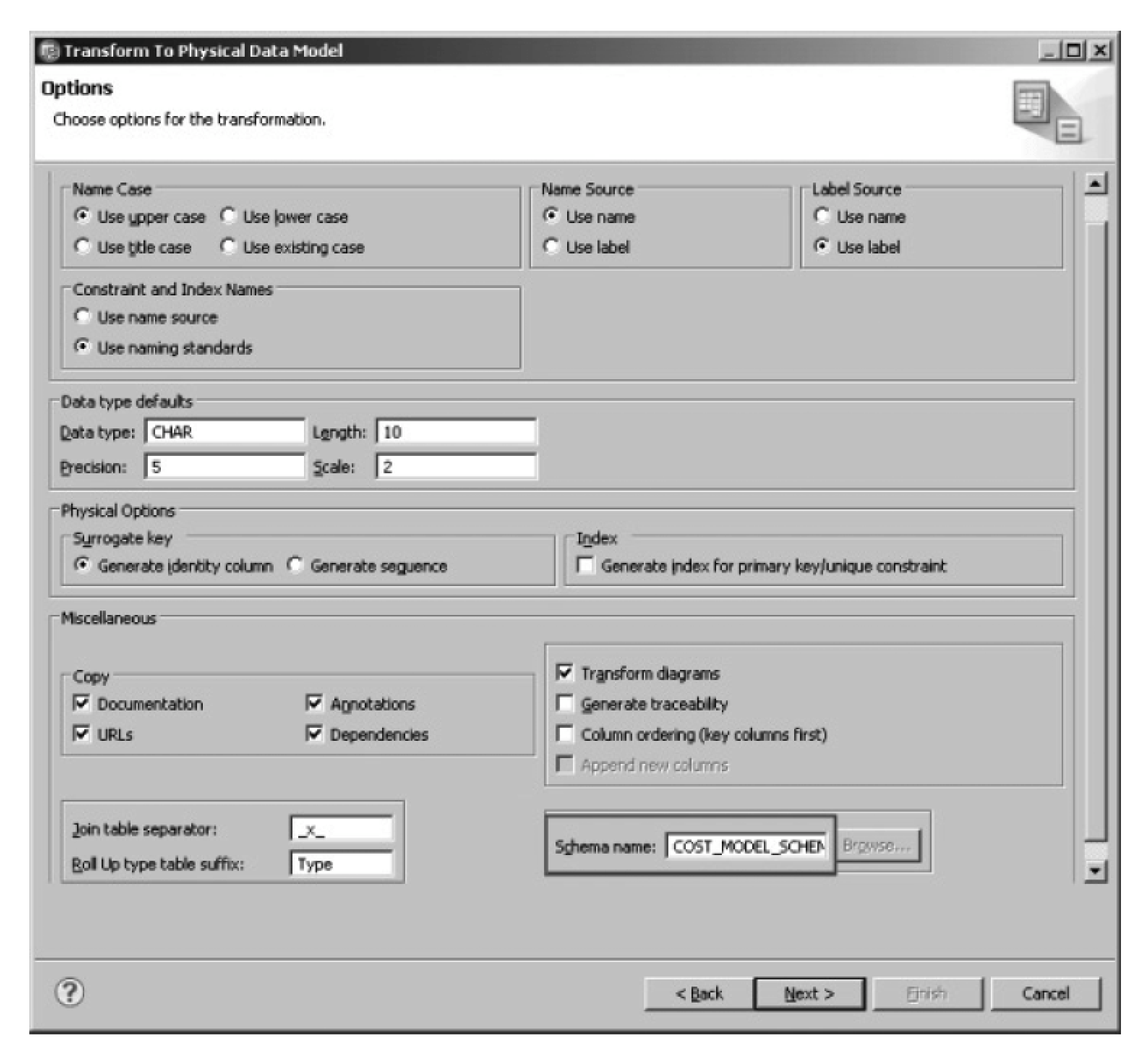


图 4.11　选项页面

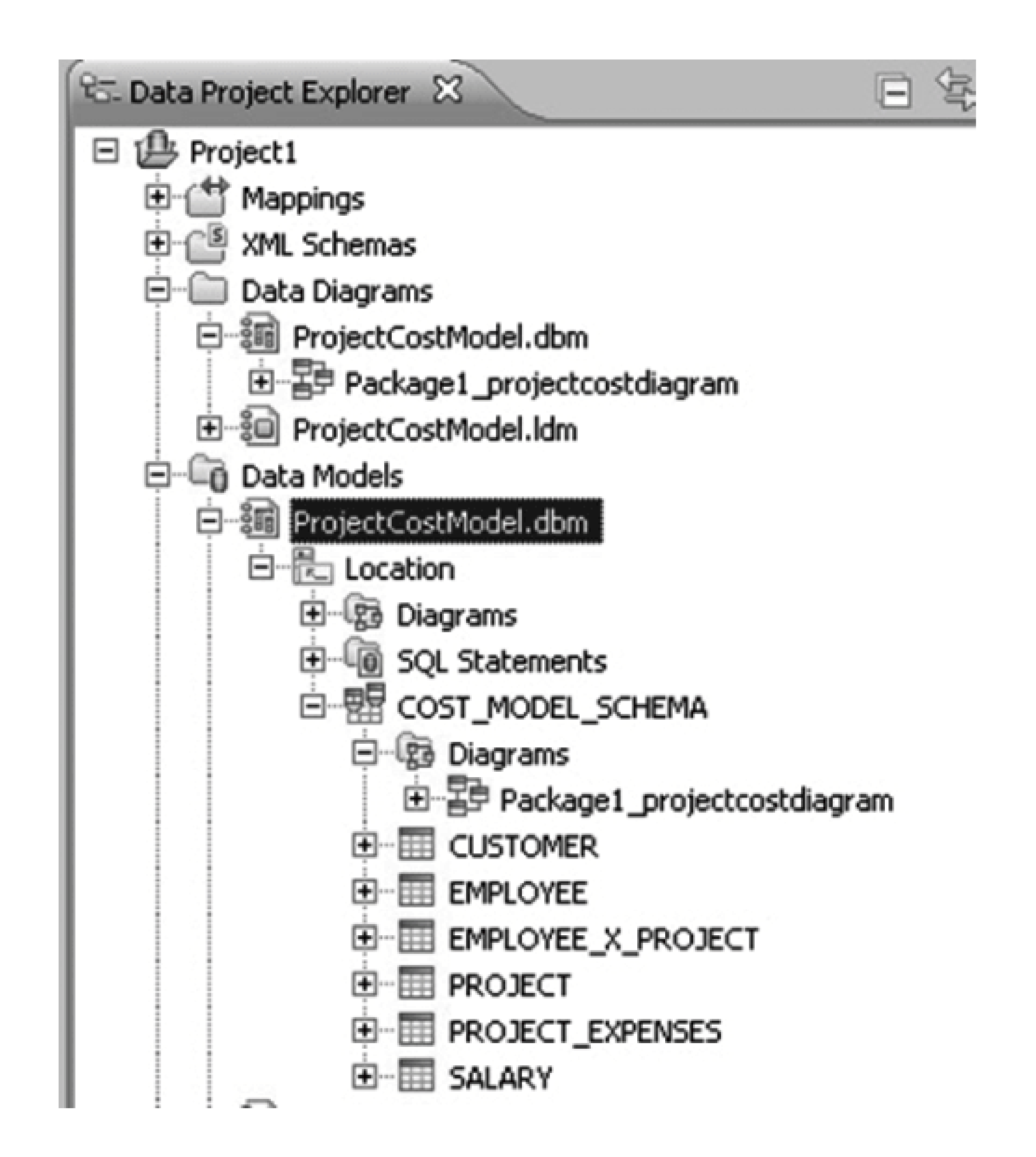


图 4.12 查看 Data Project Explorer 中的物理数据模型

在查看物理数据模型的过程中，会发现下列项是真的。

COST_MODEL_SCHEMA 模式已创建。

已经创建了 6 个表：CUSTOMER、EMPLOYEE、EMPLOYEE_X_PROJECT、PROJECT、PROJECT_EXPENSES 和 SALARY，这些表的列是从逻辑数据模型属性中转换而来的。

对于分配给员工的每个项目，员工都有一个开始日期和一个结束日期。因此，需要将 DATE 类的 START_DATE 和 END_DATE 列添加到 EMPLOYEE_X_PROJECT 表。

物理逻辑模型的对话框视图如图 4.13 所示。

(5) 创建全局临时表支持

报表需求需要使用从多个表中获取的已处理过的数据。这类聚合数据和处理过的数据通常位于要生成的报表之间。在这种情况下，数据可能被多次获取、聚合和处理，从而导致报表生成时间增加。

这种情况通过可以使用已创建的全局临时表来避免，该表将存储这类常见的聚合数据和已处理数据，这些数据是许多报表暂时需要的。

对于当前需求，所有报表都需要了解项目的各种成本类别下的费用。这些数据可计算一次，然后存储在临时表中，供所有报表重用。

项目费用可分成薪金和非薪金(原始材料、制造、包装和运输)费用。

项目的薪金费用是使用 EMPLOYEE_X_PROJECT 和 SALARY 表中的数据进行计算的。如果一个员工在某个月做了多个项目，那么薪金将根据该员工这个月进行每个项目的天数按比例分摊到单个项目中。

非薪金项目存储在 PROJECT_EXPENSES 表中。同一个成本类别下的多个费用项被聚合和存储到一个临时表中。

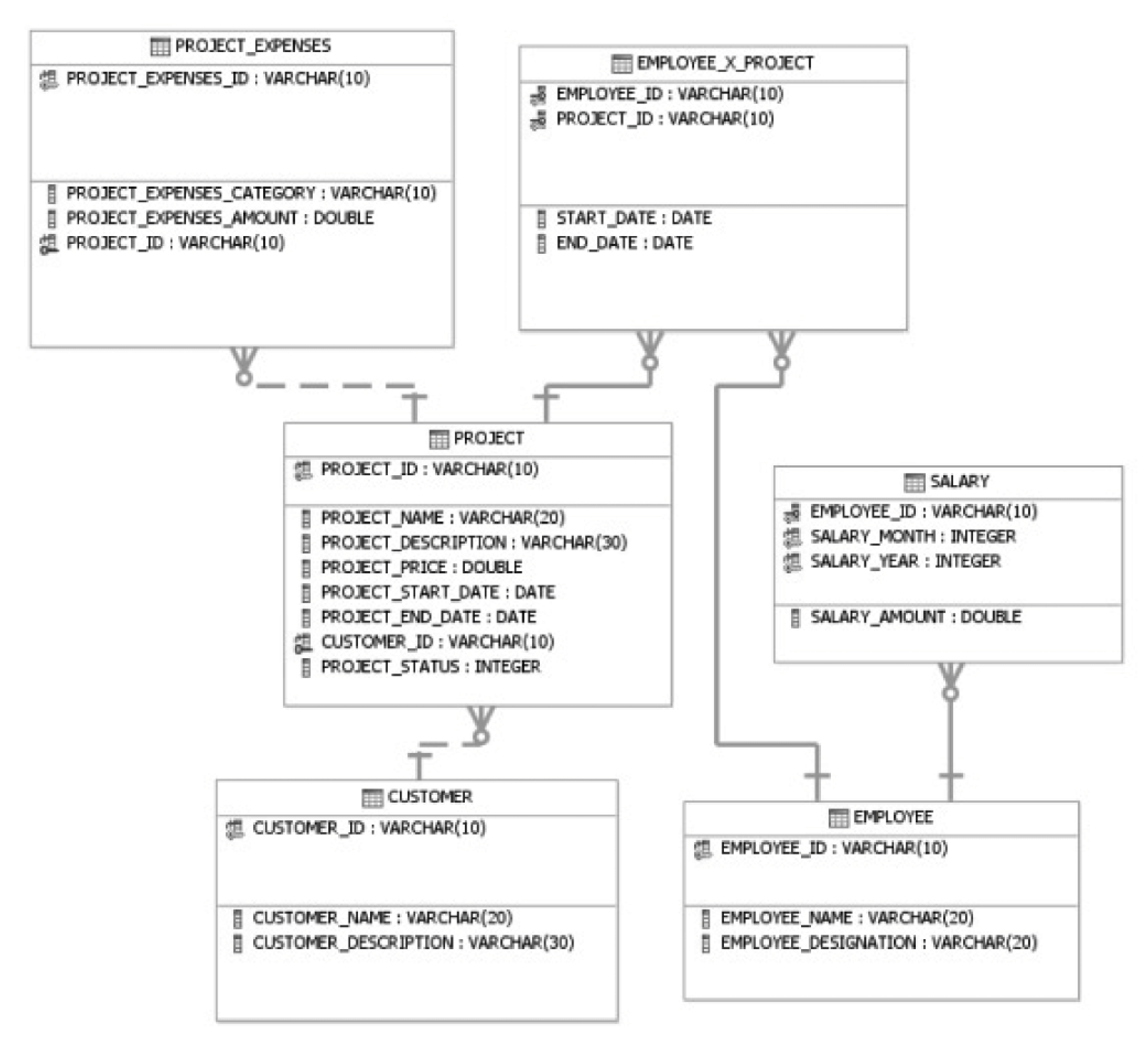


图 4.13　物理数据模型的对话框视图

通过将下列数据存储在一个临时表中，可以看到报表生成响应时间大大减少了。

- 项目薪金总额
- 项目的已发生费用的总额
- 不同成本种类的项目费用

可以创建两个已创建的全局临时表来保存上述数据：

- TEMP_PROJECT_COST_SUMMARY

PROJECT_ID：项目的唯一标识符代码。

SALARY_COST：该项目已发生的薪金费用。

TOTAL_COST：该项目已发生的费用总额。

PROJECT_PRICE：该项目的价格。

- TEMP_PROJECT_EXPENSES

PROJECT_ID：项目的唯一标识符代码。

PROJECT_EXPENSES_CATEGORY：除薪金外的费用种类。例如：原料、制造费用等。

AMOUNT：某一个费用类下的项目费用总额。这是该项目某个成本类别下的所有已发生费用的总和。

可以执行以下任务，创建一个 CGTT 来存储该常用数据。

① 在 Data Project Explorer 中右击选择 COST_MODEL_SCHEMA。

② 选择 Add Data Object→Created Global Temporary Table 命令，如图 4.14 所示。

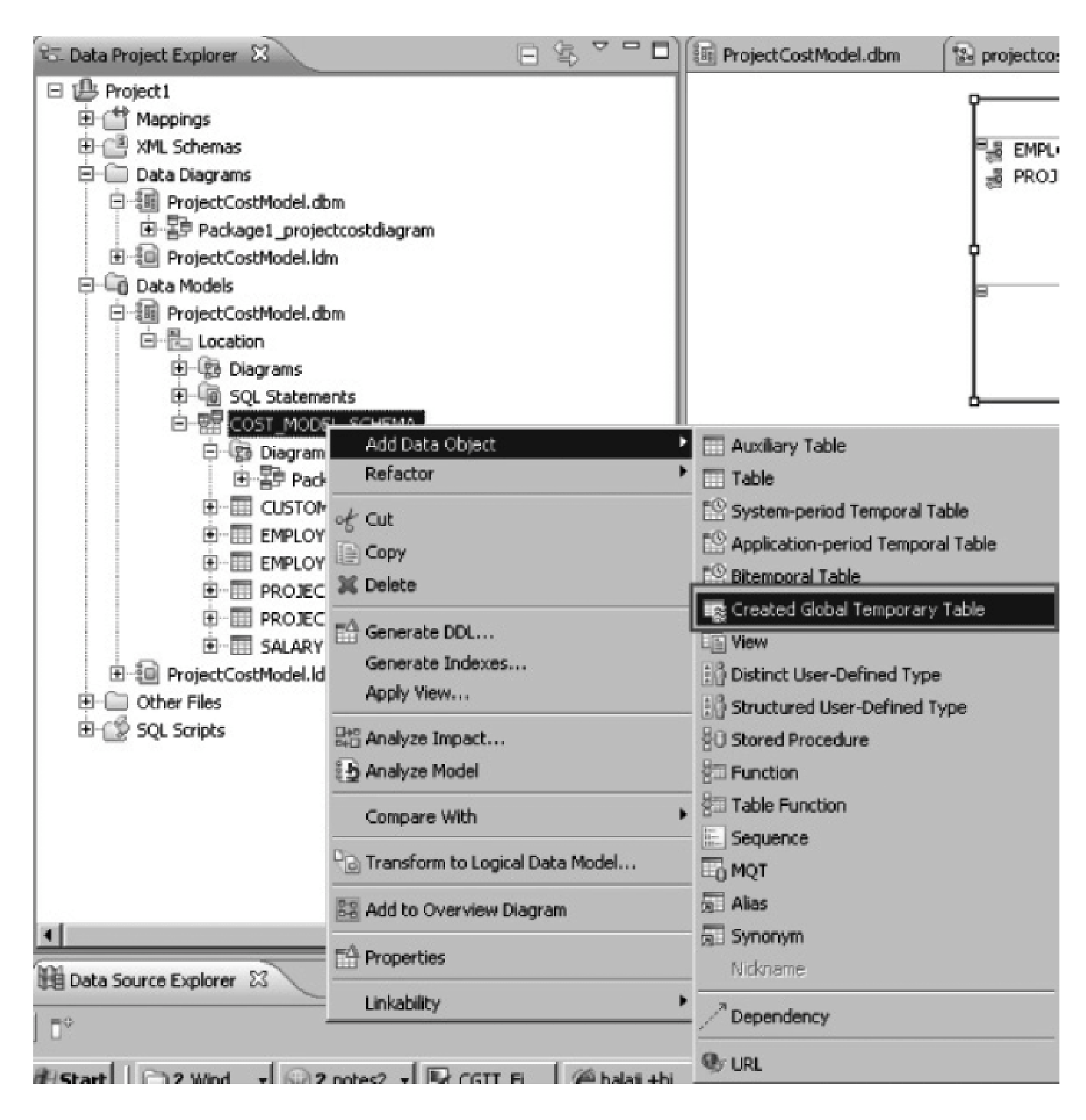


图 4.14　添加 CGTT 支持

③ 将表重命名为 TEMP_PROJECT_COST_SUMMARY。

④ 重复第 1 步和第 2 步。

⑤ 将表重命名为 TEMP_PROJECT_EXPENSES。

⑥ 对于 DB2 for Linux，UNIX，and Windows，可以为 CGTT 显式设置 On Commit 和 Loggin 属性，如图 4.15 所示。

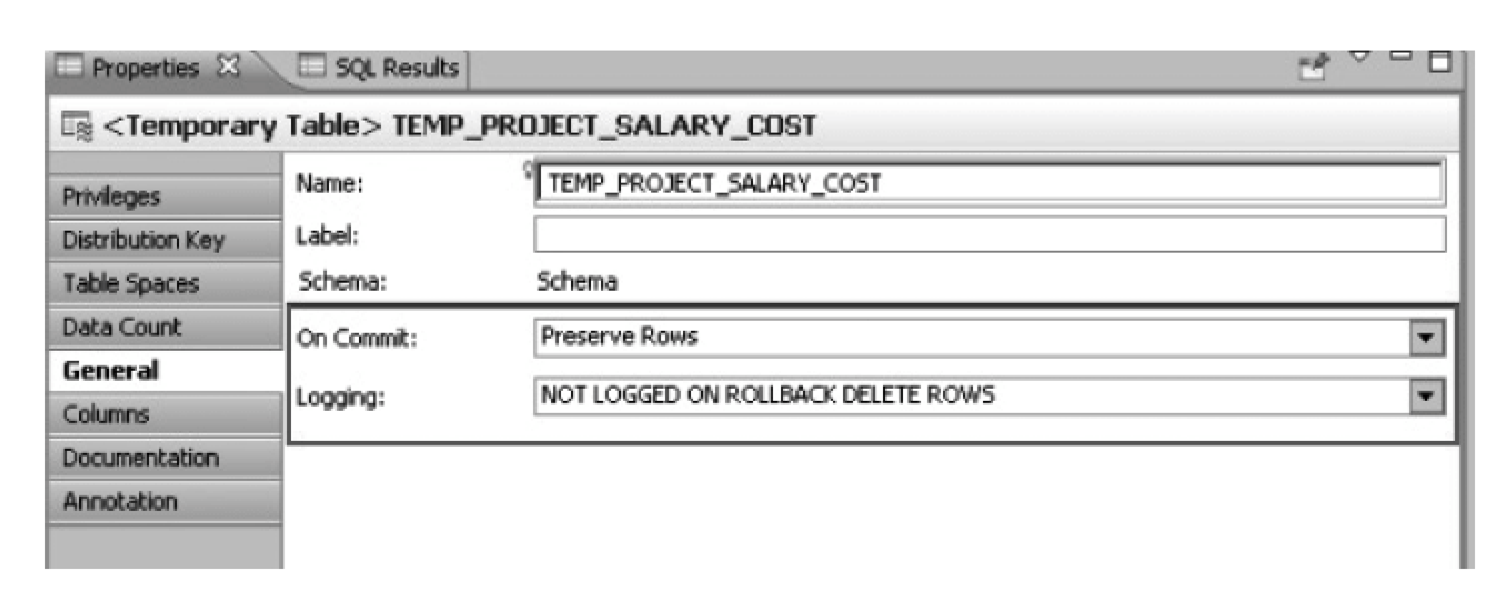


图 4.15　DB2 Linux，UNIX，and Windows 的 CGTT 属性页面

⑦ 将以下列添加到表 TEMP_PROJECT_COST_SUMMARY：PROJECT_ID、SALARY_COST、TOTAL_COST、PROJECT_PRICE，如图 4.16 所示。

⑧ 将以下列添加到表 TEMP_PROJECT_EXPENSES：PROJECT_ID、PROJECT_EXPENSES_CATEGORY 和 AMOUNT，如图 4.16 所示。

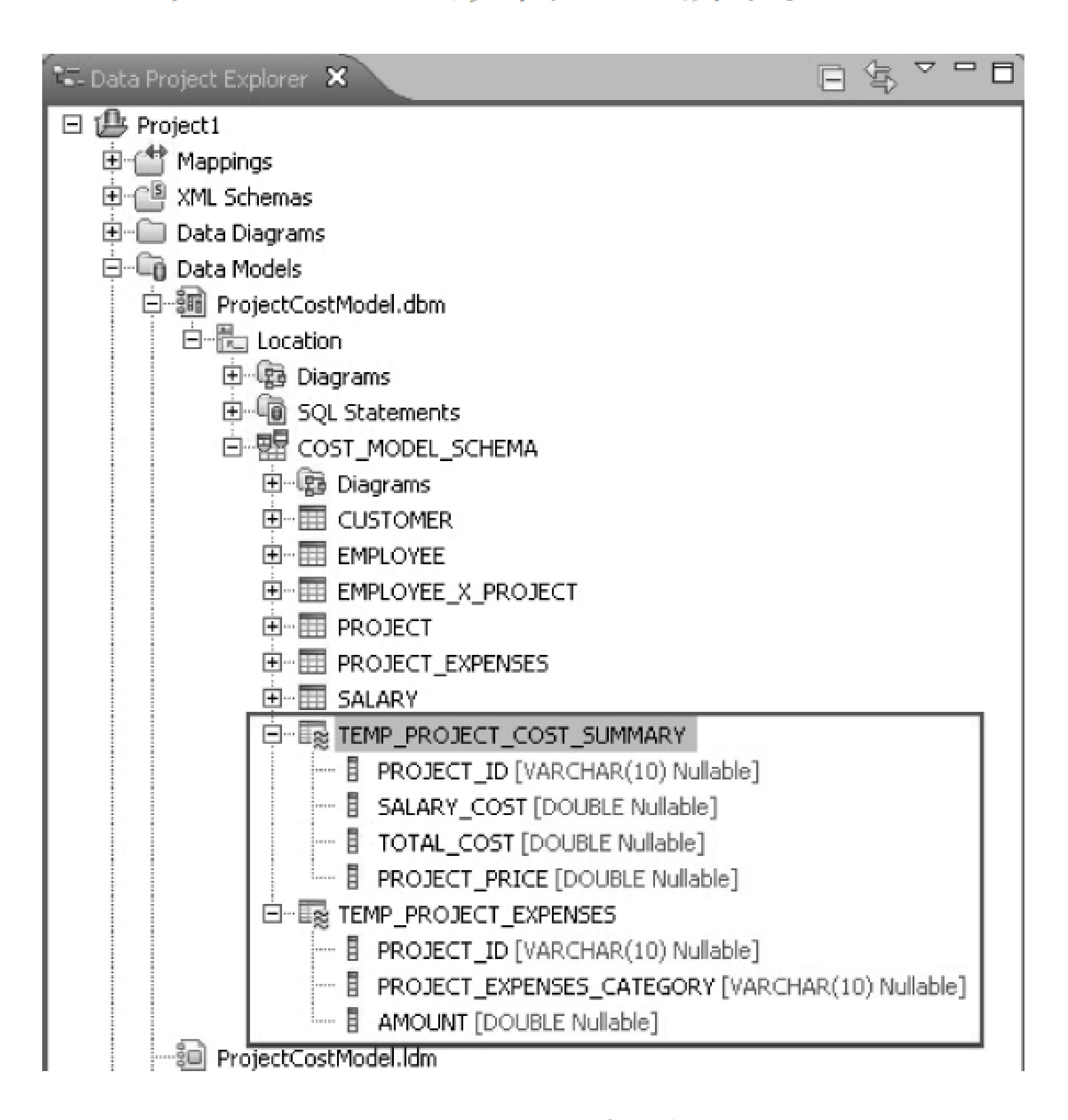


图 4.16　添加列

(6) 生成 DDL

执行以下步骤，生成 DDL 脚本来部署模型，然后与数据库管理员共享该脚本，以便在数据库服务器上运行。

① 在 Data Project Explorer 中右击选择 COST_MODEL_SCHEMA 模式。

② 选择 Generate DDL，如图 4.17 所示。

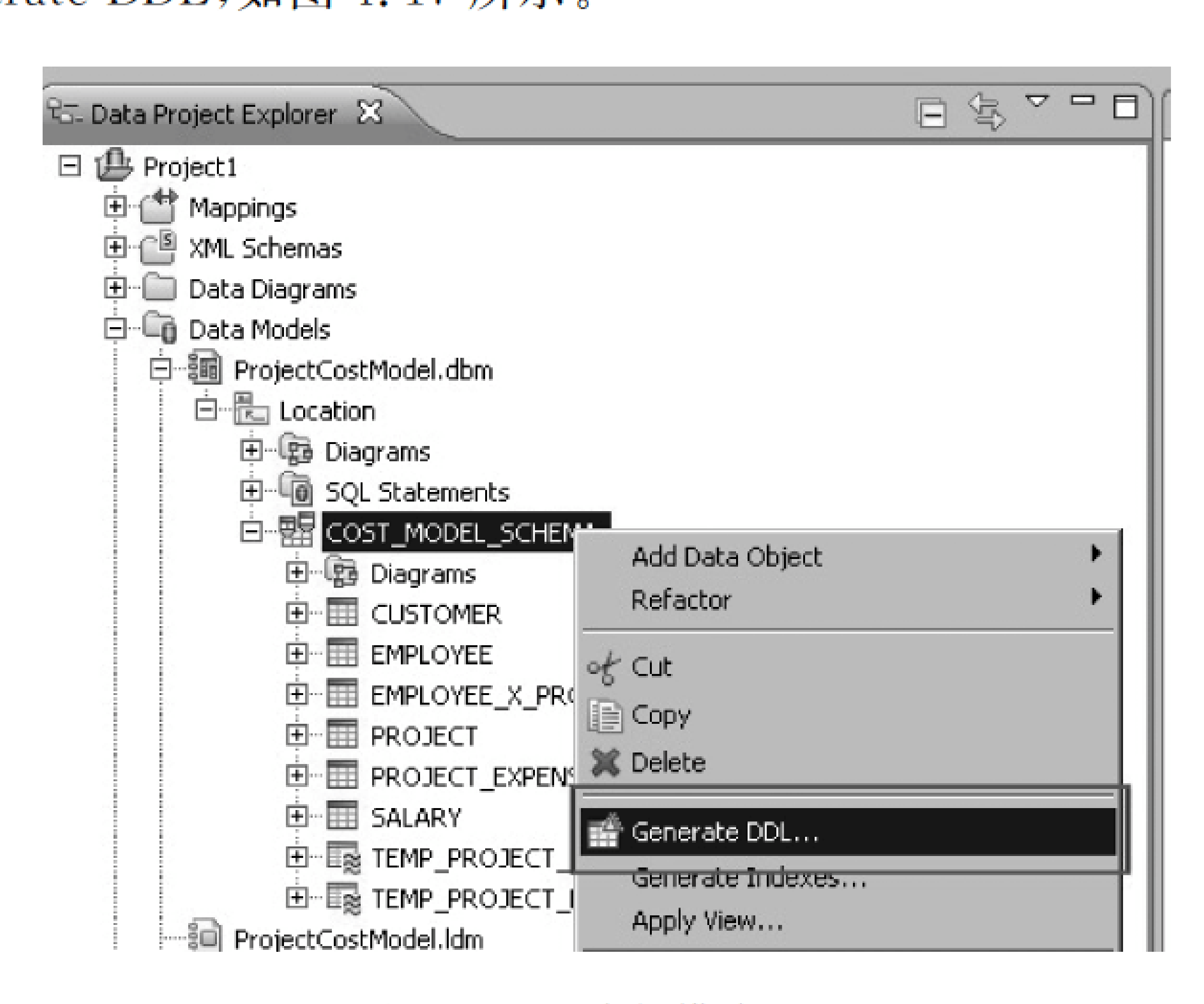


图 4.17　生成新模式 DDL

③ 完成 Generate DDL 向导页面。不要改变 Options 和 Objects 页面的默认设置。

④ 在 Save and Run DDL 页面中指定一个文件名，如图 4.18(DB2 for z/OS)和图 4.19(DB2 for Linux，UNIX，and Windows)所示。记住该文件名，以便能够与数据库管理员轻松地共享该文件，让他们可以做所需要的更改。

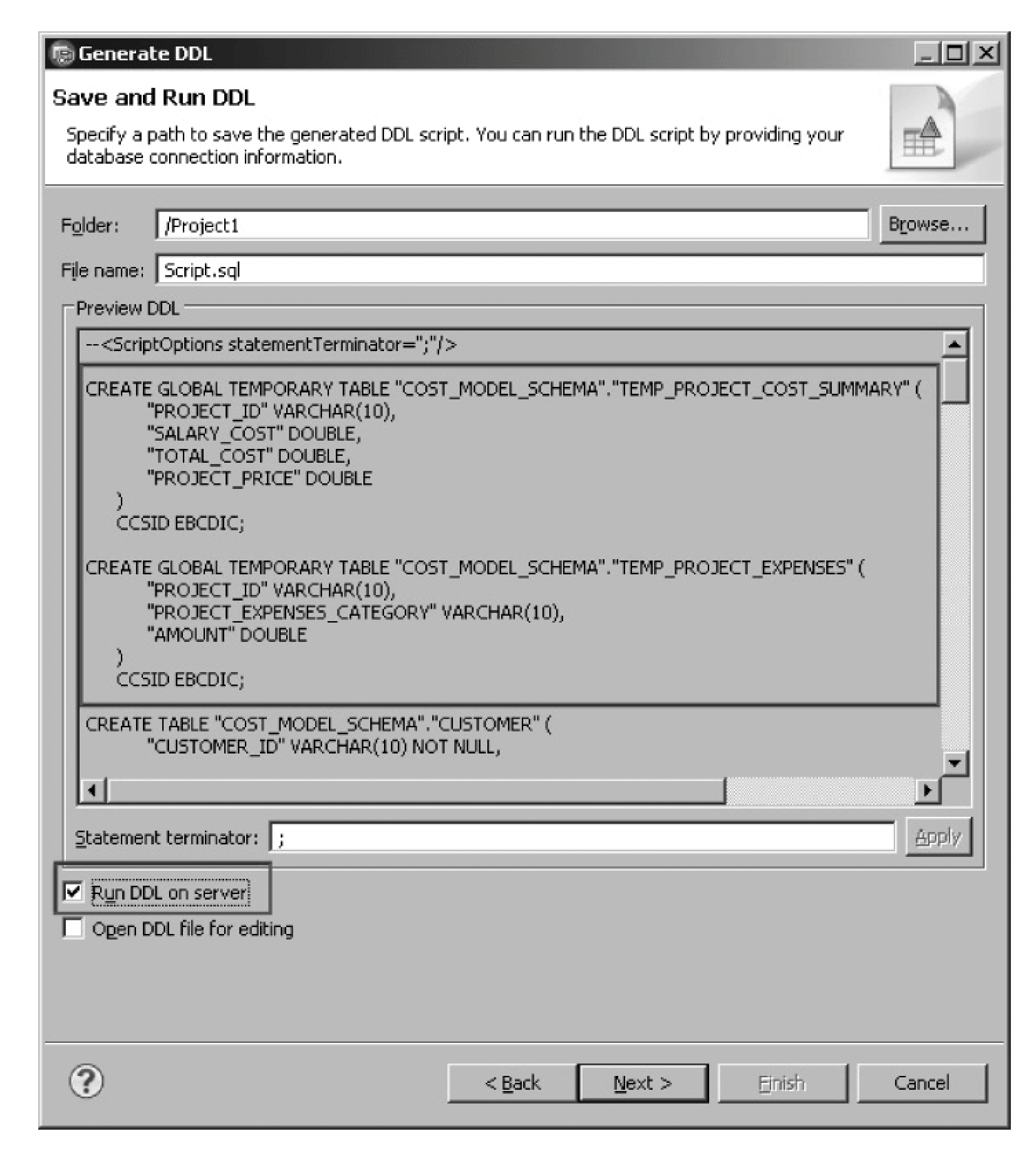


图 4.18　DB2 for z/OS 的 Save and Run DDL 页面

⑤ 如果允许直接将数据模型部署到服务器上，那么可以在选择 Next 之前选择 Run DDL on Server 选项。

⑥ 在 Select Connection 页面选择一个现有链接，如图 4.20 所示，或者通过选择 New 指定一个新连接。

⑦ 验证在页面上选择的选项，然后选择 Finish 生成 DDL。

(7) 检查数据库服务器中部署的模型

在数据库管理员运行了 DDL 脚本之后，可以通过将文件夹扩展到 COST_MODEL_SCHEMA 模式来检查 Data Source Explorer 中部署的更改。通过查看该模型，验证表 CUSTOMER、EMPLOYEE、EMPLOYEE_X_PROJECT、PROJECT、PROJECT_EXPENSES 和 SALARY 是否位于 Tables 文件夹中，TEMP_PROJECT_COST_SUMMARY 和 TEMP_PROJECT_EXPENSES 是否位于 Temporary Tables 文件夹中，如图 4.21 所示。

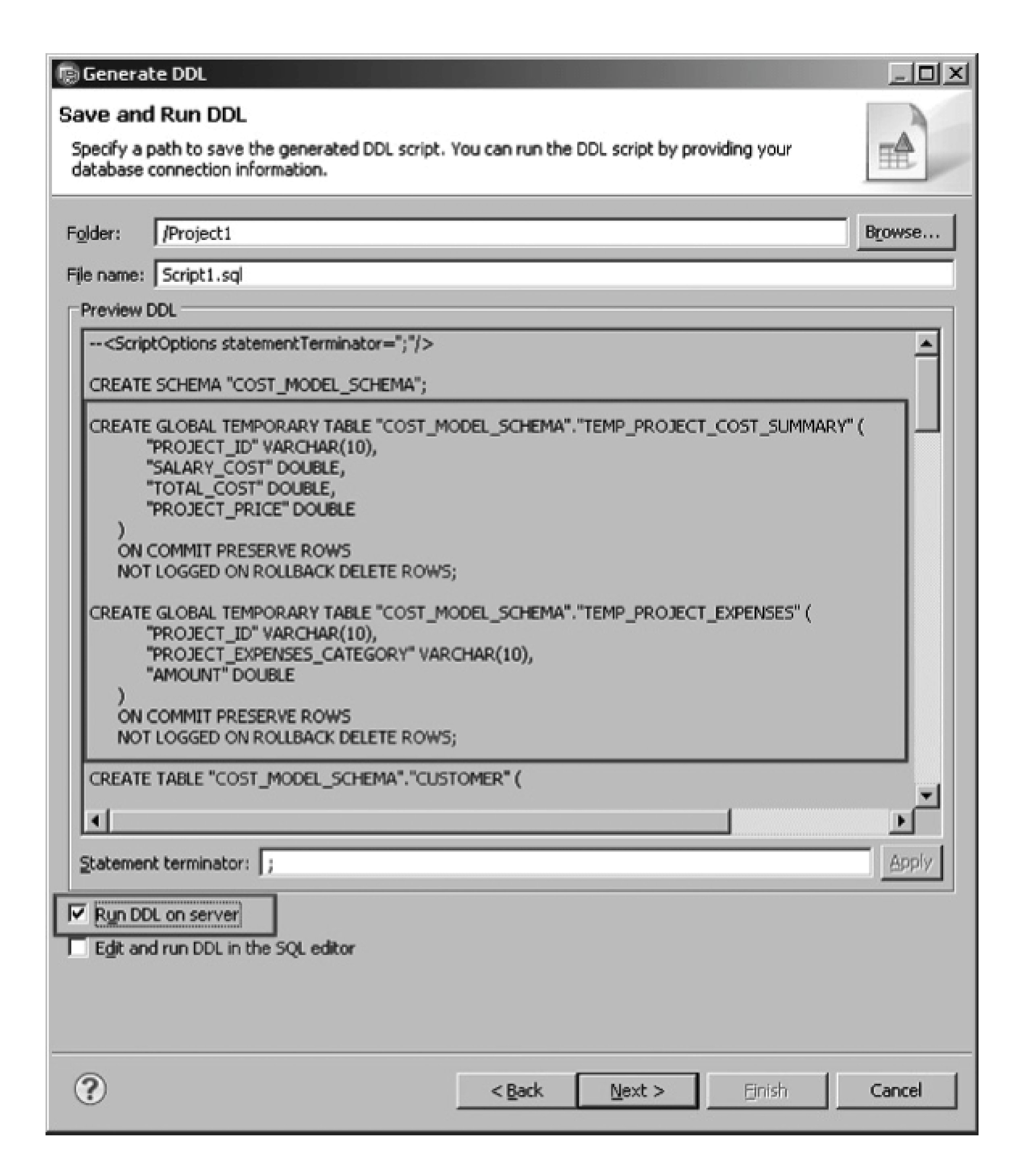


图 4.19 DB2 for Linux,UNIX,and Windows 的 Save and Run DDL 页面

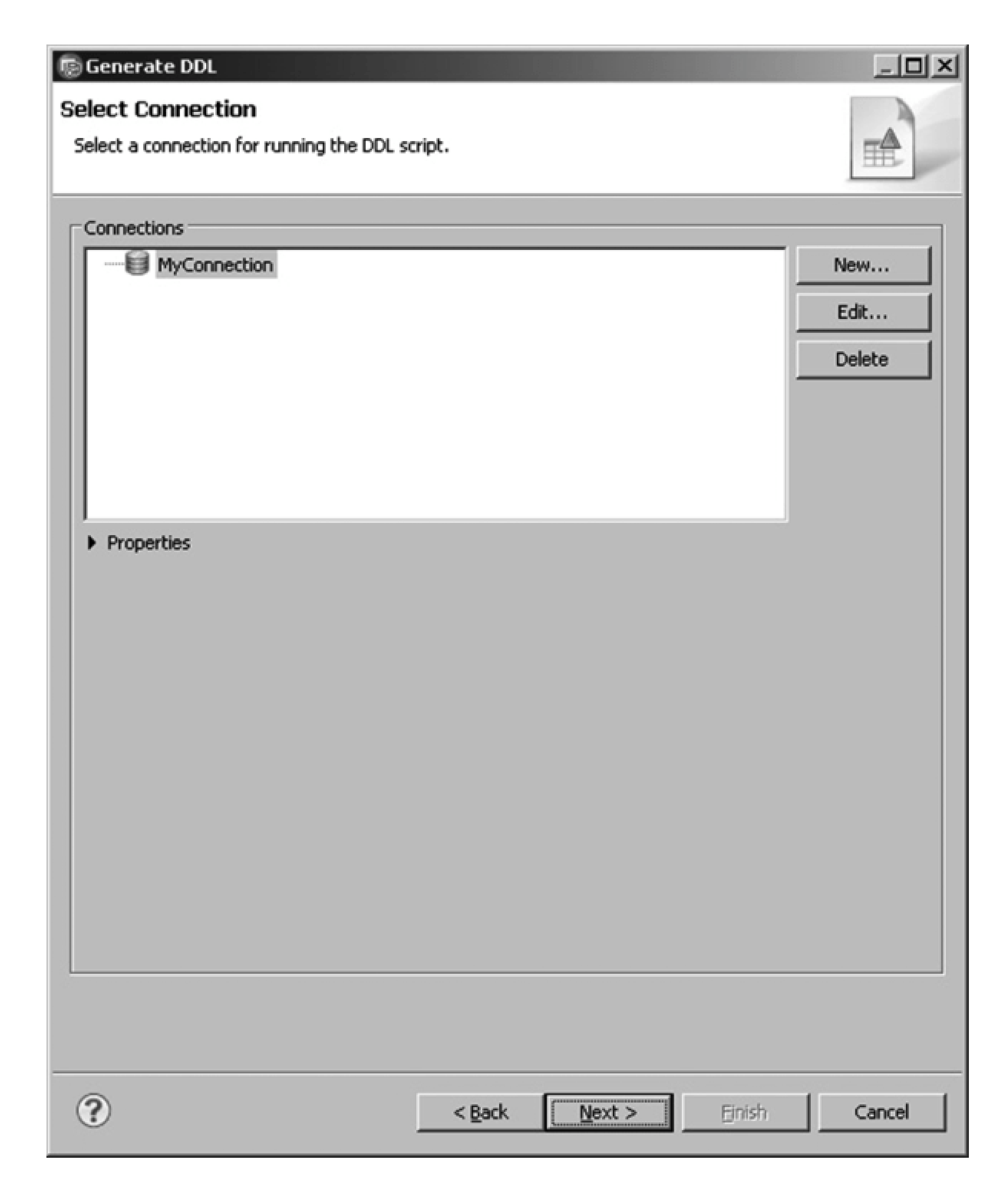


图 4.20 选择一个链接

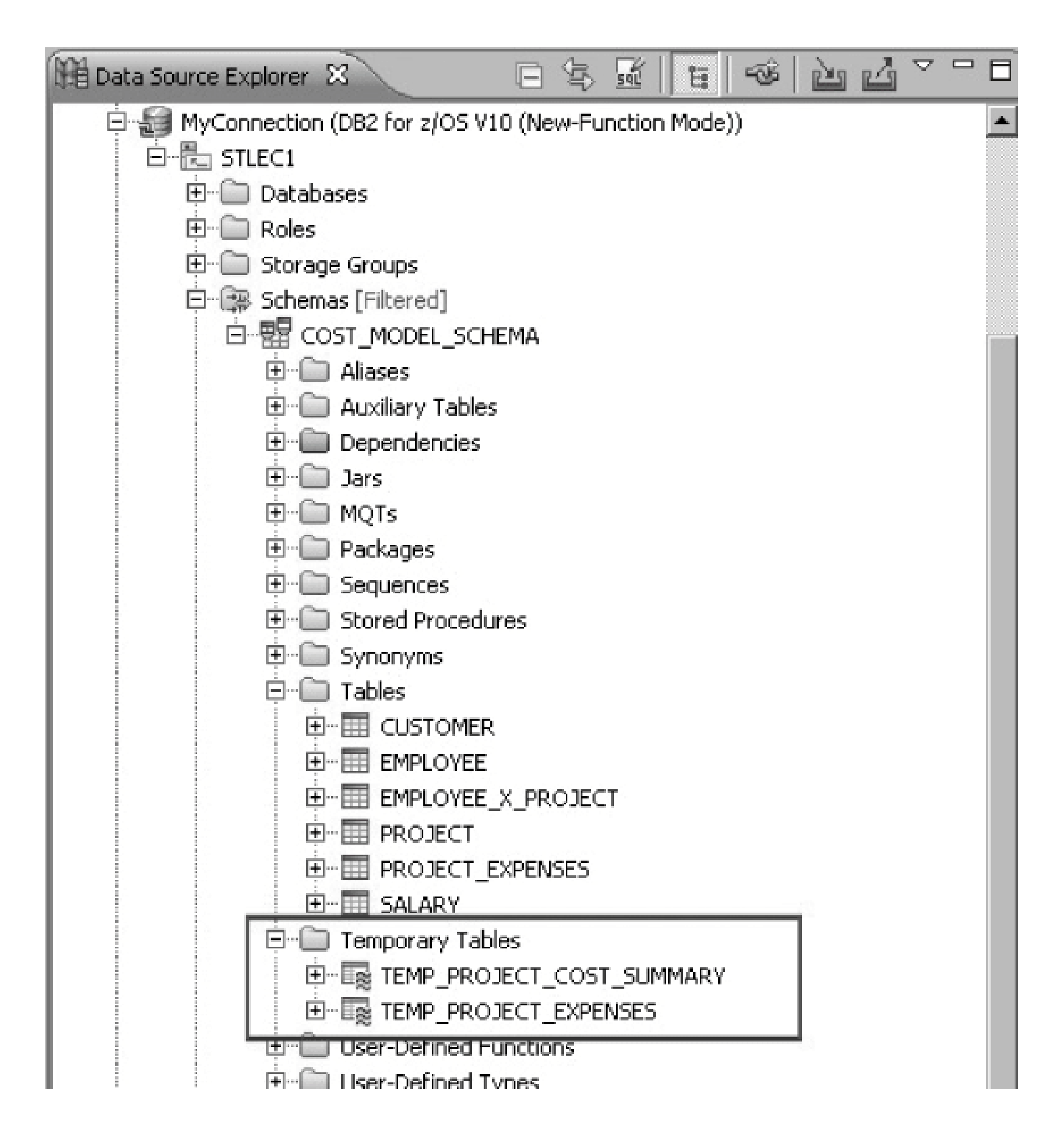


图 4.21　查看部署的项目成本模型

(8) 结束语

在本小节中,学习了如何为一个已经为某个项目成本控制系统创建的全球临时表需求进行建模。还学习了 InfoSphere Data Architect 8.5 如何为 DB2 for z/OS 和 DB2 for Linux,UNIX,and Windows 提供一个建模临时数据需求的便利方式。

这里可以轻松地创建一个具有 CGTT 支持的物理数据模型。然后,还可以使用临时特定表语法生成 DDL 脚本,该语法是将这些变更部署到目标数据库所必需的。

4.3　数据模型管理

4.3.1　数据模型管理器

数据模型管理器包括 MDM Server for PIM 系统中所有数据建模、安全建模和流程建模的组件,以及一些产品管理所需的辅助工具。

该模块的目标用户是业务流程分析专家、IT 集成专家。

用户通过数据模型管理器,可以进行独立于数据内容的元数据建模,进行基于工作流的业务流程建模,通过该模块,用户还可以定义组件级别的安全管理机制,定义待调度的工作、警报等产品管理组件。

通过图 4.22,可以看到该模块包括很多非常重要的子模块,这里会重点描述其中的规范、工作流和安全管理子模块的功能,其余子模块的功能简介可以参阅 MDM Server for PIM 产品在线用户手册。



图 4.22 数据模型管理器的子功能模块

4.3.2 规范

规范是 MDM Server for PIM 产品中最重要的组件，是对该产品中产品目录、产品分类层次、组织层次结构、待导入/导出的文件格式的元数据描述。它是所有产品信息管理活动的起始点。

规范种类分为 7 种，如图 4.23 所示，以下给出 4 种常用规范类型的简介，对于其他几种规范的使用细节，可以参阅 MDM Server for PIM 产品的在线用户手册。



图 4.23 不同类型的规范

- 文件规范(File Spec)：文件规范是用来定义待导入的文件格式的规范。举例来说，假设一个外部的 csv 文件含有 3 列，分别是产品条目 ID，产品条目名称，产品条目描述。那么为了将该文件中的产品条目信息导入到 MDM Server for PIM 产品中，就需要首先定义一个文件规范，该文件对应 csv 文件的结构描述，会含有 3 个属性节点，分别是整数类型的产品 ID、字符串类型的产品名称和字符串类型的产品描述。
- 目标规范(Destination Sepc)：与文件规范相反，目标规范是用来定义待导出的产品信息文件结构的。
- 主要规范(Primary Spec)：用来描述产品目录，或者是类别层次结构，还可能是组织层次结构中数据单体的属性结构集合。
- 辅助规范(Secondary Spec)：用来指定特定于分类(Category)级别的属性集合。

以啤酒产品管理的例子来阐述辅助规范的用途。

对于产品单品的按产地划分的类别层次管理，如果对于北方产区的啤酒需要额外记录其质检日期和质检员信息(所谓额外是指在主要规范中没有定义质检日期和质检员信息属性)，那么系统管理员就可以针对此种需求定义一个辅助规范，包含质检日期和质检员信息属性定义。将辅助规范应用到层次结构中的北方产区分类中。那么对于所有归类到北方产区的啤酒产品单体，就会在产品的主要规范的属性定义集合之外，多出此规范中定义的属性描述。可以看到，辅助规范是对主要规范的补充。这种功能对于产品元数据的灵活定制是非常重要的。

(1) 工作流

一个完善的产品信息管理系统，业务流程建模支持是不可缺少的一部分。MDM Server

for PIM 产品中的工作流是对企业业务流程的完美翻译和总结。

工作流的基本组成元素是工作流步骤(Workflow Step),工作流步骤有图 4.24 所示的几类。

在每个工作流步骤中,工作流创建者可以指定:

- 工作流步骤中的数据操作者即用户或者角色。
- 被操作的属性集合(Attribute Collection),操作可区分为只读、可编辑和必须提供信息(Required)三类。
- 基于数据流入、流出、超时事件编程,完成用户额外需要的一些功能。比如做数据合法性的校验等等。

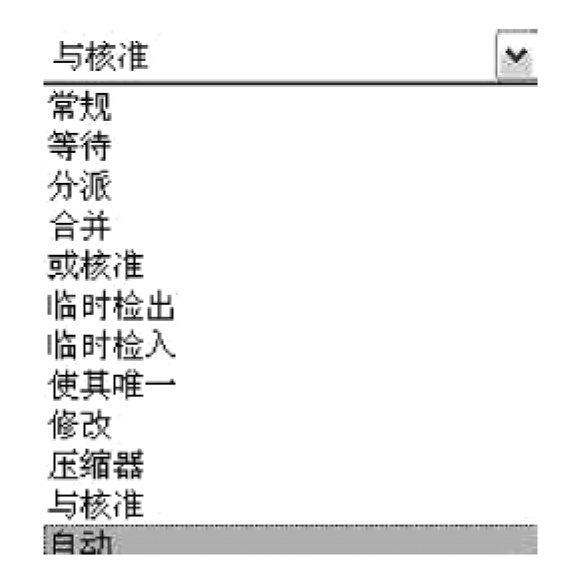


图 4.24 工作流步骤

一旦用户定义好所有的工作流步骤,就可以返回到工作流创建界面,从初始到结束将所有的工作流步骤串接起来。这样,一个完整的工作流便创建好了。以后在协作管理器中便可以用创建好的工作流作为骨架,创建协作区,将业务流程实例化,进行实际的产品数据(无论是产品条目还是产品分类)的业务流程管理。

(2) 安全管理

MDM Server for PIM 产品的安全体系所囊括的所有组成元素都是通过安全管理模块创建的。

通常,系统管理员会在系统中创建角色,如为销售经理创建一个经理角色。随后,系统管理员会创建访问控制组,在访问控制组中,MDM Server for PIM 系统中所有的对象组件类型,例如产品目录、产品条目、产品分类等等都会列出来,如图 4.25 所示。然后,用户可以针对访问控制组加入关心的角色,从而形成组件与角色的矩阵,针对该角色选择授予或拒绝特定组件的访问权限。

访问控制组详细信息	
名称	访问控制组名称
描述	访问控制组描述
目录	**角色** trigo_basic
列示	☑
编辑目录视图	☑
查看项	☑
添加项	☐
克隆项	☑
修改项	☑
删除项	☑
将项重新分类	☐
总结项	☐
导出	☐
属性	☑
差别	☑
回滚	☐
搜索	☐
删除	☐
运行预览脚本	☐
层次结构	
列示	☑
编辑层次结构视图	☑

图 4.25 访问控制组

针对上文提出的啤酒产品管理的用例，以下从数据建模、安全模型建模、业务流程建模三个方面描述数据模型管理器的常用功能。

- 数据建模：用户的需求清晰地描述了产品目录的主要规范需求。这个主要规范应该包含5个属性，产品序列号是唯一确定产品单体的主键。所以，用户可以通过规范控制台，创建如图4.26所示的主要规范。其中，产品序列号是整型，产品名称和产品的生产厂商是字符串型，产品的生产日期是日期型，核准合格是标志型(Flag，为真或为假)。

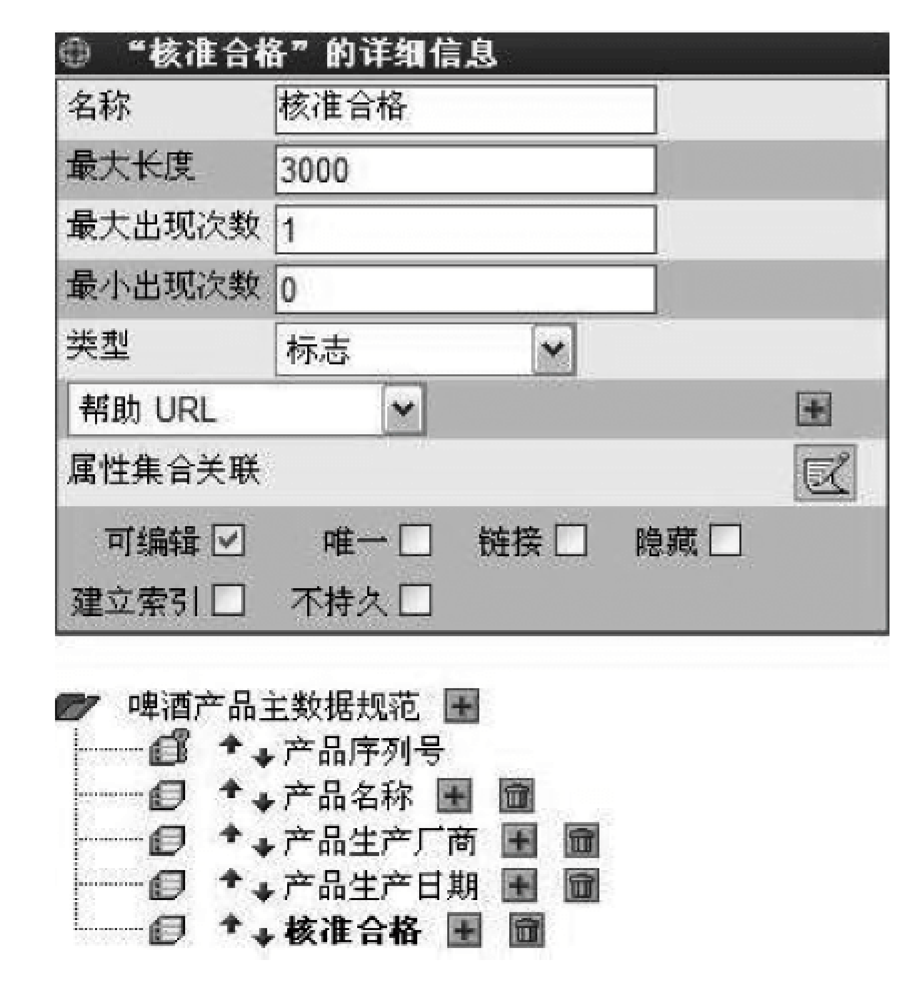


图 4.26 啤酒产品主要数据规范

根据产品的分类需求，应该创建一个主要规范描述产品分类的层次结构，创建一个辅助规范来描述属于北方产区的产品单体的质检信息。主要规范需要两个字符串型节点，地区名称和地区简介，地区名称作为主键。辅助规范也需要两个字符串型属性节点，质检日期和质检员信息，如图4.27所示。

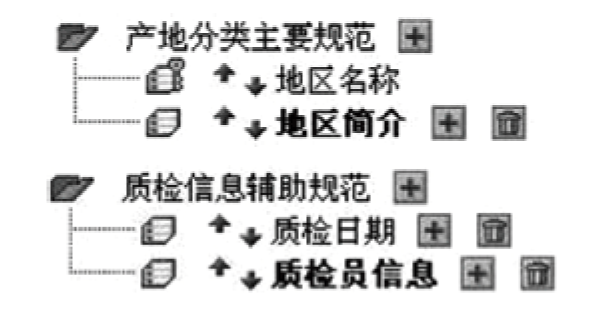


图 4.27 分类主要规范和质检辅助规范

- 安全模型建模：根据业务流程自动化管理需求中的角色和用户需求，系统管理员应该为产品经理和质检员各建立一个账户(ProManager 和 QA)。不妨授予产品经理超级管理员角色，质检员为一般管理员角色。注意，为简单起见，这里不单独为产品经理和质检员建立不同的角色，关联不同的访问控制组，是因为在协作区中规定的用户访问权限会覆盖用户所在的角色所拥有的权限。图4.28是客户经理的概要文件信息。
- 业务流程建模：针对该种业务需求，系统管理员可以创建一个工作流，该工作流包含一个"与核准"工作流步骤。产品条目从起始点进入自定义的工作流步骤"与核

用户概要文件

姓	三
名	张
用户名	ProManager
职位	产品经理
电子邮件	foo@company.com
地址	
电话	
传真	

修改用户概要文件

重新设置密码

新密码	
确认密码	

重新设置密码



图 4.28　客户经理的用户概要文件

准”，只有得到质检员（QA）以及产品经理（ProManager）的双重审核批准，目标产品条目才能通过审核步骤（这里假设产品经理和质检员要更新核准通过的值为真，代表操作者同意核准通过），进入工作流成功步骤。如果质检员或者产品经理任何一方拒绝核准通过，那么该产品条目将进入工作流失败步骤。图 4.29、图 4.30 给出工作流步骤和工作流的定义。

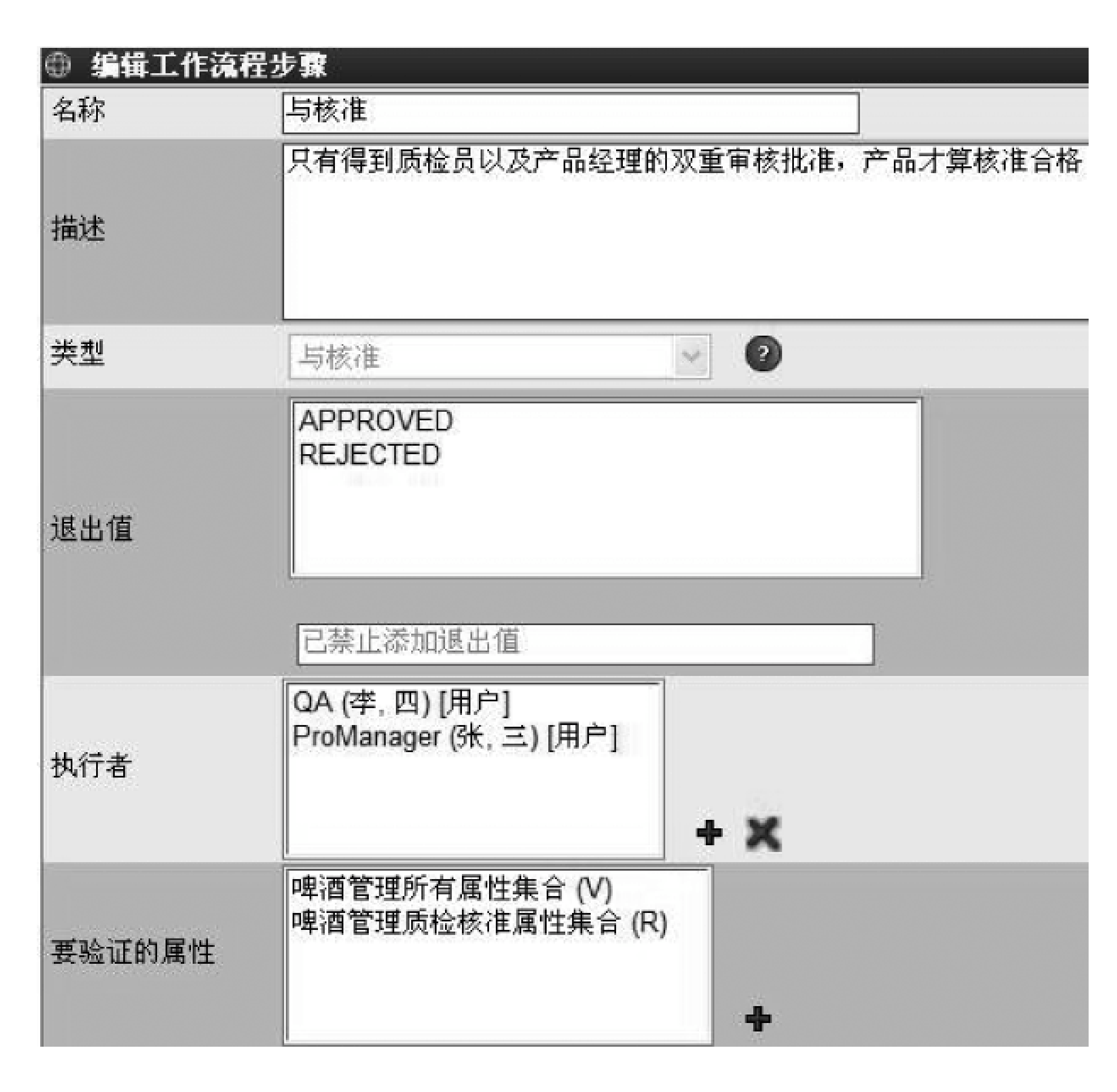


图 4.29　工作流“与核准”步骤



图 4.30　啤酒类产品管理工作流

4.4　主数据管理

主数据管理(Master Data Management,MDM)描述了一组规程、技术和解决方案,这些规程、技术和解决方案用于为所有利益相关方(如用户、应用程序、数据仓库、流程以及贸易伙伴)创建并维护业务数据的一致性、完整性、相关性和精确性。

主数据管理的关键就是“管理”。主数据管理不会创建新的数据或新的数据纵向结构。相反,它提供了一种方法,使企业能够有效地管理存储在分布系统中的数据。主数据管理使用现有的系统,它从这些系统中获取最新信息,并提供了先进的技术和流程,用于自动、准确、及时地分发和分析整个企业中的数据,并对数据进行验证。

主数据管理解决方案具有以下特性:

- 在企业层面上整合了现有纵向结构中的客户信息以及其他知识和深层次信息。
- 共享所有系统中的数据,使之成为一系列以客户为中心的业务流程和服务。
- 实现对于客户、产品和供应商都通用的主数据形式,加速数据输入、检索和分析。
- 支持数据的多用户管理,包括限制某些用户添加、更新或查看维护主数据的流程的能力。
- 集成产品信息管理、客户关系管理、客户数据集成以及可对主数据进行分析的其他解决方案。

由于和主数据管理关联的方法和流程的运行与企业的业务流系统及其他系统彼此独立,因此这些方法和流程不仅能检索、更新和分发数据,还能满足主数据的各种用途。主数据管理通过将数据与操作应用程序实时集成来支持操作用途。主数据管理还通过使用经过

授权的流程来创建、定义和同步主数据来支持协作用途。最后，主数据管理通过事件管理工具事先将主数据推送至分析应用程序来支持分析用途。

4.4.1 数据管理的范畴和主数据管理的概念

企业数据管理的内容及范畴通常包括交易数据、主数据以及元数据，如图 4.31 所示。

- 交易数据。用于记录业务事件，如客户的订单、投诉记录、客服申请等，它往往用于描述在某一个时间点上业务系统发生的行为。
- 主数据。主数据则定义企业核心业务对象，如客户、产品、地址等，与交易流水信息不同，主数据一旦被记录到数据库中，需要经常对其进行维护，从而确保其时效性和准确性；主数据还包括关系数据，用以描述主数据之间的关系，如客户与产品的关系、产品与地域的关系、客户与客户的关系、产品与产品的关系等。
- 元数据。即关于数据的数据，用以描述数据类型、数据定义、约束、数据关系、数据所处的系统等信息。

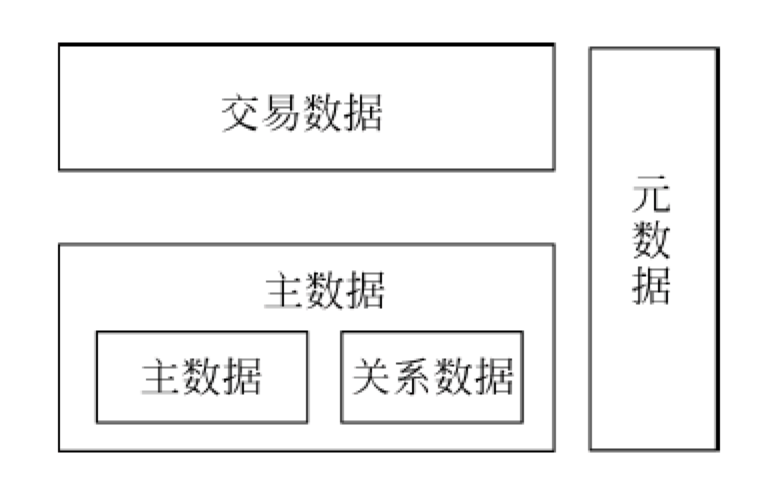


图 4.31 数据管理的范畴

主数据管理是指一整套的用于生成和维护企业主数据的规范、技术和方案，以保证主数据的完整性、一致性和准确性，如图 4.32 所示。主数据管理的典型应用有—客户数据管理(Customer Data Integration)和产品数据管理(Product Information Integration)。

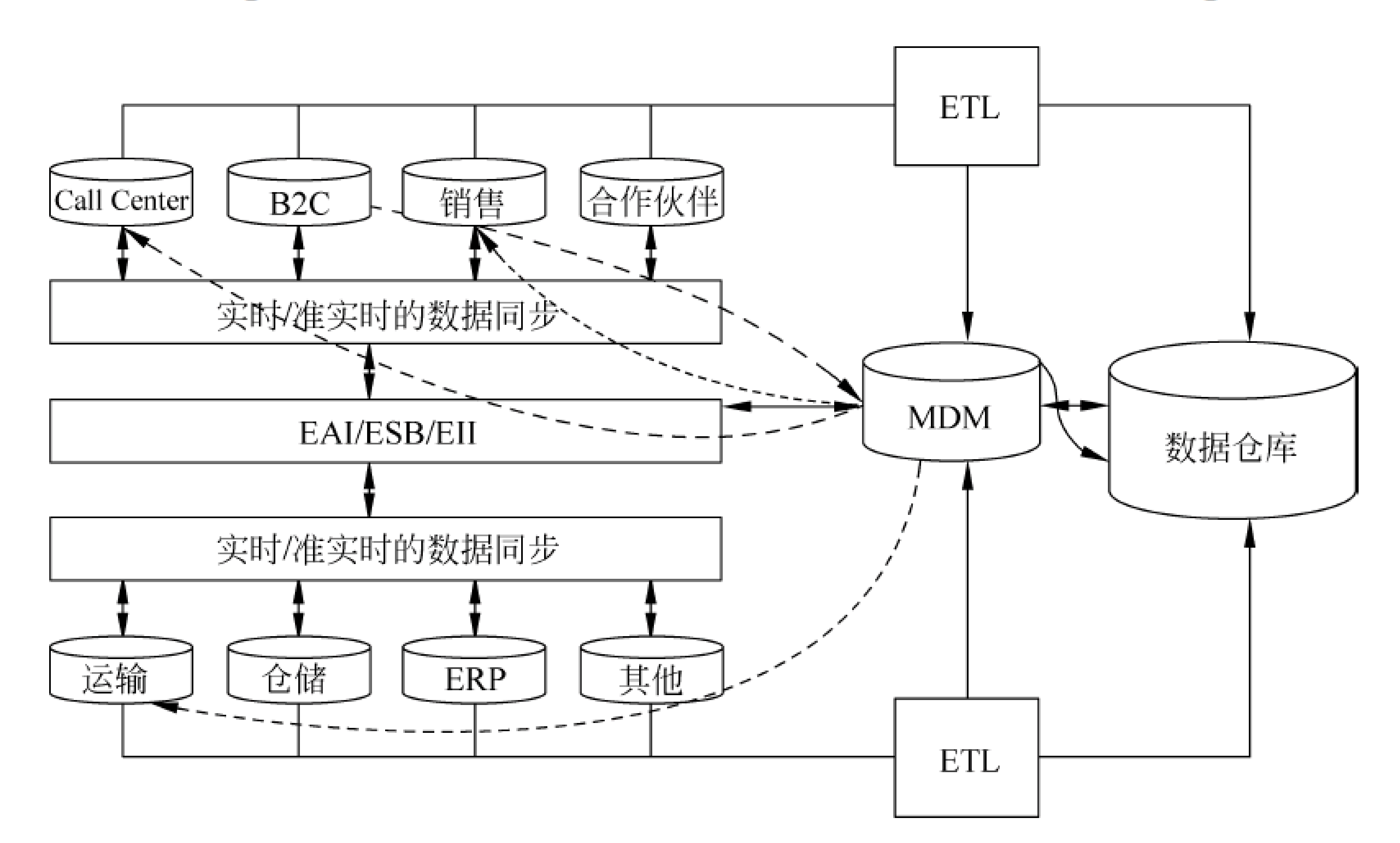


图 4.32 主数据管理的信息流

一般来说，主数据管理系统从 IT 建设的角度而言都会是一个相对复杂的系统，它往往会和企业数据仓库/决策支持系统以及企业内的各个业务系统发生关系，技术实现上也会涉及 ETL、EAI、EII 等多个方面，如图 4.32 所示，一个典型的主数据管理的信息流为：

- 某个业务系统触发对企业主数据的改动。
- 主数据管理系统将整合之后完整、准确的主数据分发给所有有关的应用系统。
- 主数据管理系统为决策支持和数据仓库系统提供准确的数据源。

因此对于主数据管理系统的建设，要从建设初期就考虑整体的平台框架和技术实现。

以客户主数据为例，常见的主数据域包括：

- 参与方(Party)。参与方包含的范围是所有与企业发生了或者发生过正式业务关系的任何合法的实体，比如填写了投保单的参与方。参与方是分类别的，可以是个人、机构和团体。对于参与方来说，因为开展业务的需要，可能要对他们进行分级、分类，比如 VIP，黑名单等。个人包括个人基本属性、个人名称、职业、性别、教育等自然属性；机构是指在法律上有登记的组织实体，可以分为政府机构、商业机构、非盈利机构等类别；团体可以有多种形态，比如他们可以是家庭、兴趣小组、某个大机构中的一部分，或者通过某种数据分析技术得出的客户细分群体。
- 参与方在业务中扮演的角色(Party Role)。例如，对于保险行业而言，可以有：投保人，被保人，受益人，担保人，报案人，核保人，查勘员，核赔人等。
- 参与方之间的关系(Relationship)。例如可以是：夫妻关系、父子关系、母女关系、兄弟姐妹关系、总(母)公司分(子)公司关系、企业事业单位隶属、上下级关系等。
- 账户(Account)。账户是客户使用企业服务的付费实体。
- 地址(Location)。它记录的是每个参与方可能拥有的所有联系地址，地址的类别包括邮寄地址、E-mail 地址、电信联络地址等。
- 合同(Contract)。参与方与企业之间的契约。

主数据有几个鲜明的特点，其中包括：它是准确的、集成的，它是跨业务部门的，是在各个业务部门被重复使用的。

4.4.2 主数据管理的意义

数据的集成、共享、质量、治理是主数据管理的四大要素，如图 4.33 所示，主数据管理要做的就是从企业的多个业务系统中整合最核心的、最需要共享的数据(主数据)，集中进行数据的清洗和丰富，并且以服务的方式把统一的、完整的、准确的、具有权威性的主数据分发给全企业范围内需要使用这些数据的操作型应用和分析型应用，包括各个业务系统、业务流程和决策支持系统等。

主数据管理使得企业能够集中化管理数据，在分散的系统间保证主数据的一致性，改进数据合规性，快速部署新应用，充分了解客户，加速推出新产品的速度。从 IT 建设的角度，主数据管理可以增强 IT 结构的灵活性，构建覆盖整个企业范围内的数据管理基础和相应规范，并且更灵活地适应企业业务需求的变化。

以客户主数据为例，客户主数据是目前企业级客户普遍面临的一个问题，在大多数企业中，客户信息通常分散于 CRM 等各个业务系统中，而每个业务系统中都只有客户信息的片

断，即不完整的客户信息，但却缺乏企业级的完整、统一的单一客户视图，结果导致企业不能完全了解客户，无法协调统一的市场行为，导致客户满意度下降，市场份额减少。因此，建立客户主数据系统的目的在于：

- 整合并存储所有业务系统和渠道的客户及潜在客户的信息：一方面从相关系统中抽取客户信息，并完成客户信息的清洗和整合工作，建立企业级的客户统一视图；另一方面，客户主数据管理系统将形成的统一客户信息以广播的形式同步到其他各个系统，从而确保客户信息的一致。
- 为相关的应用系统提供联机交易支持，提供客户信息的唯一访问入口点，为所有应用系统提供及时和全面的客户信息；服务于 OCRM 系统，充分利用数据的价值，在所有客户接触点上提供更多具有附加价值的服务。
- 实现 SOA 的体系结构：建立客户主数据系统之前，数据被锁定在每一个应用系统和流程中，建立主数据管理系统之后，数据从应用系统中被释放出来，并且被处理成为一组可重用的服务，被各个应用系统调用。

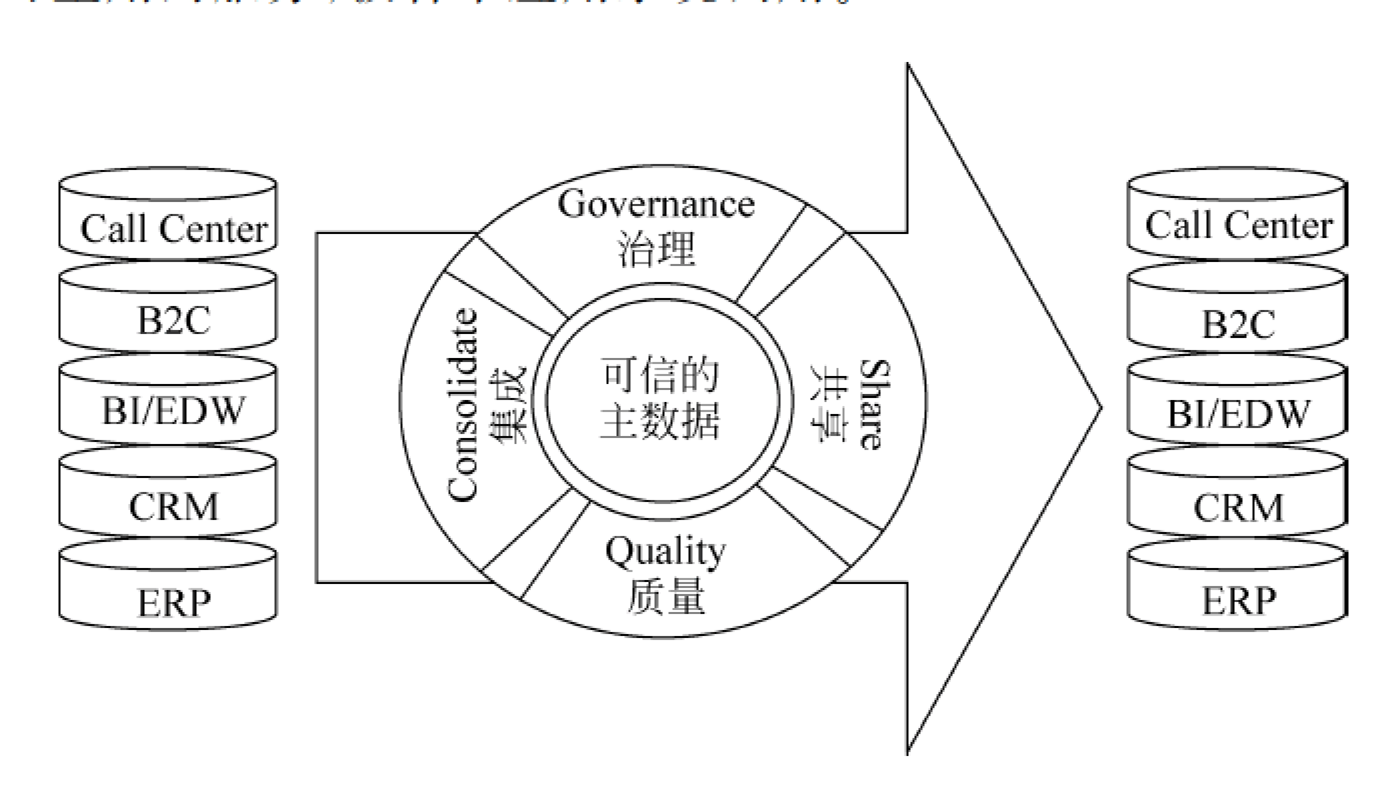


图 4.33 主数据管理的要素

4.4.3 主数据管理系统与数据仓库系统的关系

主数据管理系统与数据仓库系统是相辅相成的两个系统，但二者绝不是重复的，也不是互斥的。它们有很多共同之处：

- 首先，二者对企业都具有相同的价值，可以减少数据冗余和不一致性、提升对数据的洞察力，二者都是跨部门的集中式系统。
- 其次，二者都依赖很多相同的技术手段，都会涉及 ETL 技术，都需要元数据管理，都强调数据质量。
- 第三，就是二者建设手段类似，都需要数据治理的规范作为指导，都需要不同系统、不同部门的协作，需要统一的安全策略。

但是，主数据管理系统和数据仓库/决策支持系统二者之间也存在很多不同：

- 处理类型不同。主数据管理（MDM）系统是偏交易型的系统，它为各个业务系统提供联机交易服务，系统的服务对象是呼叫中心、B2C、CRM 等业务系统；而数据仓库是属于分析型的系统，面向的是分析型的应用，是在大量历史交易数据的基础上进行多维分析，系统的使用对象是各层领导和业务分析、市场销售预测人员等。

- 实时性不同。与传统的数据仓库方案的批量 ETL 方式不同，主数据管理系统在数据初始加载阶段要使用 ETL，但在后续运行中要大量依赖实时整合的方式来进行主数据的集成和同步。
- 数据量不同。数据仓库存储的是大量的历史数据和各个维度的汇总数据，可能会是海量的，而 MDM 存储的仅仅是客户和产品等信息。

虽然主数据管理系统和数据仓库系统异同共存，但是二者却有着紧密的联系，并且可以互相促进、互为补充。举例而言，数据仓库系统的分析结果可以作为衍生数据输入到 MDM 系统，从而使 MDM 系统能够更好地为操作型 CRM 系统服务。以航空公司为例，客户的主数据模型大致可以分为三部分：

首先包括客户基本信息和偏好信息。

- 客户基本信息
  - 个人及公司信息
  - 消费者市场状况
  - 常旅客会员卡号，状态，及累计里程等
  - 客户间关系(个体-个体，个体-公司)
  - 联系地址，包括电话、电子邮件等
- 客户偏好信息
  - 餐食偏好
  - 是否吸烟
  - 座位偏好
  - 机型偏好
  - 公务舱位偏好
  - 旅行舱位偏好
  - 休息室服务偏好

除了这两部分之外，还可以从数据仓库系统中提取相关的信息，作为客户主数据的衍生信息部分，从而更好地、全方位地描述客户特征。

- 衍生信息
  - 本月飞行里程
  - 年度飞行里程(最近 12 个月内)
  - 提前预订倾向
  - 习惯预订模式
  - 使用自主服务倾向
  - 上次预订使用的信用卡号
  - 累计/本月转签/取消航班次数
  - 转签航班倾向
  - 取消航班倾向
  - No Show 倾向等

4.4.4 主数据管理系统和 ODS 的关系

在某些情况下，主数据管理系统和 ODS 系统可能容易被混淆，的确，从实时上来看，主数据管理系统和 ODS 系统存储的都是实时数据，但是二者存储的数据内容是全然不同的，主数据管理系统中不存储交易数据，比如银行客户的交易流水信息是不应该放在主数据管理系统中进行管理的，这是 MDM 与 ODS 的一个很大区别。举一个航空公司的例子，比如某个客户在电子商务网站上订了一张机票，产生一个订单，然后他又通过呼叫中心要求改签，这个场景中，两个系统之间要实现客户信息和订单信息的共享，其中客户信息共享通过 MDM 系统来实现，而订单信息则需要采用 ODS 或其他手段进行共享，这里是不推荐把此类信息交由 MDM 系统来管理的。

4.4.5 主数据管理解决方案介绍

目前业界比较常见的主数据管理解决方案主要可以分为 3 类：

- 第一是依托专业套装软件来实现主数据管理，这类方案是作为套装软件的一部分，主要是为套装软件的其他模块提供服务的，因此，通常功能都缺乏完善性。
- 还有一类是侧重于分析型应用的主数据管理，这类方案在数据实时同步以及面向交易型应用时通常缺乏整体方案的完整性。
- 再有一类就是专注于主数据管理的中立的、完整的解决方案，这一类应用独立于套装软件，不仅具有整体架构的完整性和先进性，从功能上讲往往也最为完善，除了具有比较完整的数据模型(Data Model)之外，还会提供广泛的集成性，具备先进的机制实现数据同步，并且可以对外提供多种预置的主数据服务被外部交易系统调用，从而使系统具有很强的实时操作性，同时还强调主数据管理、主数据质量控制以及主数据维护的手段和规范性。

4.4.6 企业主数据管理系统逻辑架构

在一个完整的主数据管理解决方案中，除了主数据管理的核心服务组件之外通常还会涉及企业元数据管理、企业信息集成、ETL、数据分析和数据仓库以及 EAI/ESB 等其他各种技术和服务组件。

其中主数据管理服务又包括如下一些主要的服务组件。

- Interface Services：为企业中需要主数据的所有业务系统提供各种服务接口，通过实时的、批量的接口可以读取或者修改主数据，这些接口包括 Batch，Web Services，XML Interface，Messaging Interface，Publish/Subscribe，Import/Export Services，Data Standardization Interface，Directory Integration 等。除了这些标准的技术接口之外，对于某些专有系统还提供适配器(Adapter)接口，通过适配器接口可以和一些特有的系统做接口，例如企业中的传统(Legacy)应用系统或者 SAP 等打包应用。
- Lifecycle Management Services：履行针对主数据的 CRUD 操作，执行对主数据存储库中的数据进行更新、存取和管理时的业务逻辑，除此之外，它还负责维护主数据的衍生信息，例如客户之间的关系、客户的偏好、客户在各种客户服务渠道上的行为轨迹等。Lifecycle Management Services 贯穿整个主数据管理的生命周期，它利用

Data Quality Management Services 来确保数据质量、利用 Master Data Event Management Services 来捕获各种主数据变化等相关的事件，以及利用 Hierarchy and Relationship Management Services 用来维护数据实体之间的关系和层次。

- Data Quality Management Services：确保主数据的质量和标准化，这在主数据管理解决方案中是一个非常重要的组件，在从各个业务系统获取数据之后，要对数据进行清洗和验证，例如对于地址而言，要弥补地址的缺失、地市的缺失、邮编的缺失、进行地址的标准化等。对于其他数据要进行非空检查、外键检查、数据过滤等。然后要对数据进行匹配/重复识别、自动进行基于规则的合并/去重、交叉验证等，并且还要遵从企业的数据管控规范和流程。它可以是 Master Data Management Services 的一个内部组件，也可以调用整个企业的 Information Integrity Services 来实现。
- Authoring Services：依据数据管控流程，定义和扩展企业的主数据模型。
- Hierarchy Relationship and Management Services：定义数据实体的层次(Hierarchy)、分组(Grouping)、关系(Relationship)、版本(Version)等。
- Master Data Event Management Services：捕获事件并且触发相应的操作，包括事件发现、事件管理和通知功能，它在主数据管理系统和业务系统之间进行数据同步时起到至关重要的作用。
- Base Services：提供通用服务，包括安全控制、错误处理、交易日志、事件日志等功能。
- Master Data Repository：主数据存储库，包括 Metadata，Master Data，History Data，Reference Data 等。

下面介绍两个这些逻辑组件之间的协作场景：

场景 1：初始数据加载

① 源数据从外部业务系统及 EDW 系统中通过批处理方式复制到磁带；

② 数据被加载到 Staging DB，进行数据质量分析；

③ Data Quality Management Services 对数据进行清洗、匹配、标准化等；

④ ETL Transform and Load Services 对合格数据进行转换并准备好加载数据；

⑤ Master Data Interface Services 接收批处理更新请求，调用 Lifecycle Management Update Service 进行数据的批量更新；

⑥ Lifecycle Management Update Service 调用 Hierarchy & Relationship Management Services 和 Base Services 更新主数据库。

场景 2：主数据库更新，然后同步到各业务系统

① 某业务系统发起一个创建主数据的交易，该业务系统将交易数据以消息的形式发送到消息队列；

② MDM Interface Services 捕获该消息，进行消息解析，并调用 Security and Privacy Services 进行权限验证；

③ MDM Interface Services 调用 Lifecycle Mgmt. Update Service；

④ Lifecycle Mgmt. Update Service 再调用 Data Quality Management Services 进行数据的清洗和标准化；

⑤ Update Service 调用 Search Services 发现该主数据已经存在，确认这是对已有主数

据的更新操作;

⑥ Update Service 通过调用外部系统对数据进行扩充;

⑦ Update Service 在更新主数据库之前调用 Event Management Services;

⑧ Event Management Services 确认是否需要涉及数据管控方面的处理;

⑨ Update Service 调用 Hierarchy & Relationship Management Services 并且更新主数据库;

⑩ Audit Logging Services 记录相应交易日志和历史数据;

⑪ MDM Lifecycle Management Service 调用 MDM Interface Services 返回更新处理请求;

⑫ 源业务系统接收到处理请求之后,利用 MDM 系统发回来的数据对本地的应用系统数据库进行更新操作。

其他所有需要主动被更新的相关业务系统都会接收到更新后的最新数据。

4.5 元数据管理

4.5.1 明确元数据管理策略

在最开始的时候,元数据(Meta Data)是指描述数据的数据,通常由信息结构的描述组成,随着技术的发展,元数据内涵有了非常大的扩展,比如 UML 模型、数据交易规则、用 Java、.NET、C++等编写的 APIs、业务流程和工作流模型、产品配置描述和调优参数以及各种业务规则、术语和定义等。在大数据时代,元数据还应该包括对各种新数据类型的描述,如对位置、名字、用户点击次数、音频、视频、图片、各种无线感知设备数据和各种监控设备数据等的描述等。元数据通常分为业务元数据、技术元数据和操作元数据等。业务元数据主要包括业务规则、定义、术语、术语表、运算法则和系统使用业务语言等,主要使用者是业务用户。技术元数据主要用来定义信息供应链(Information Supply Chain,ISC)各类组成部分元数据结构,具体包括各个系统表和字段结构、属性、出处、依赖性等,以及存储过程、函数、序列等各种对象。操作元数据是指应用程序运行信息,比如其频率、记录数以及各个组件的分析和其他统计信息等。

从整个企业层面来说,各种工具软件和应用程序越来越复杂,相互依存度逐年增加,相应的追踪整个信息供应链各组件之间数据流动、了解数据元素含义和上下文的需求越来越强烈。在从应用议程往信息议程的转变过程中,元数据管理也逐渐从局部存储和管理转向共享。从总量上来看,整个企业的元数据越来越多,仅现有的数据模型中就包含了成千上万的表,同时还有更多的模型等着上线,随着大数据时代的来临,企业需要处理的数据类型越来越多。为了企业更高效地运转,企业需要明确元数据管理策略和元数据集成体系结构,依托成熟的方法论和工具实现元数据管理,并有步骤地提升其元数据管理成熟度。

为了实现大数据治理,构建智慧的分析洞察,企业需要实现贯穿整个企业的元数据集成,建立完整且一致的元数据管理策略,该策略不仅仅针对某个数据仓库项目、业务分析项目、某个大数据项目或某个应用单独制定一个管理策略,而是针对整个企业构建完整的管理策略。元数据管理策略也不是技术标准或某个软件工具可以取代的,无论软件工具功能多

强大都不能完全替代一个完整一致的元数据管理策略，反而在定义元数据集成体系结构以及选购元数据管理工具之前需要定义元数据管理策略。

元数据管理策略需要明确企业元数据管理的愿景、目标、需求、约束和策略等，依据企业自身当前以及未来的需要确定要实现的元数据管理成熟度以及实现目标成熟度的路线图，完成基础本体、领域本体、任务本体和应用本体的构建，确定元数据管理的安全策略、版本控制、元数据订阅推送等。企业需要对业务术语、技术术语中的敏感数据进行标记和分类，制定相应的数据隐私保护政策，确保企业在隐私保护方面符合当地的法律法规，如果企业有跨国数据交换、元数据交换的需求，也要遵循涉及国家的法律法规要求。企业需要保证每个元数据元素在信息供应链每个组件中语义上保持一致，也就是语义等效（Semantic Equivalence）。语义等效程度可以强也可以弱，在一个元数据集成方案中，语义等效（平均）程度越强则整个方案的效率越高。语义等效的强弱程度直接影响元数据的共享和重用。

4.5.2 元数据集成体系结构

在明确了元数据管理策略后需要确定实现该管理策略所需的技术体系结构，即元数据集成体系结构。各个企业的元数据管理策略和元数据管理成熟度差别较大，因此元数据集成体系结构也多种多样。大体上元数据集成体系结构可以分为点对点的元数据集成体系结构、中央辐射式元数据体系结构、基于公共仓库元模型（Common Warehouse MetaModel，CWM）驱动的点对点元数据集成体系结构、基于 CWM 模型驱动的中央存储库元数据集成体系结构、分布式（联邦式）元数据集成体系结构和层次/星型元数据集成体系结构等。

针对信息供应链中不同的组件，为了实现跨组件的元数据交换和集成，最开始人们采用点对点的方式进行，也就是每一对组件之间通过一个独立的元数据桥（Metadata Bridge）进行元数据交换，桥一般是双向的能够理解两个方向的元数据映射。点对点的元数据集成体系结构帮助用户实现了跨企业的元数据集成和元数据交换，对提升信息化水平提供了巨大帮助。这种体系结构在应用过程中，也暴露了很多问题，比如元数据桥的构建工作量和耗时都非常大，对中间件厂商、应用厂商、集成商和用户来说都是一个巨大的挑战，而且构建元数据桥还必须具有所有者的元数据模型和接口的详细信息。构建完成的桥很多时候无法在构建其他元数据桥时进行重用，因此开发和维护费用大幅度增加，用户投资回报率（ROI）不高。以动态数据仓库为例，其点对点的元数据集成体系结构如图 4.34 所示，信息供应链各组件之间的双向箭头表示全部的数据流，实心箭头表示不同的元数据桥和与之关联的元数据流。

通过使用中央元数据存储库（Central Metadata Repository）取代各个工具软件和应用程序之间的点对点连接方式，改成中央元数据存储库与各个工具软件和应用程序实现元数据交换的访问层（也是一种桥），可以有效降低总成本，减少建立点对点元数据桥的工作，提高投资回报率。信息供应链各组件可以从存储库访问元数据，不必与其他产品进行点对点交互。这种使用中央元数据存储库进行元数据集成方式就是中央辐射式元数据体系结构（Hub-and-Spoke Meta Data Architecture），如图 4.35 所示。由于特定的元数据存储库是围绕其自身的元模型、接口和交付服务建立的，所以仍需要建立元数据桥实现与 ISC 各组件的互相访问。

采用模型驱动的元数据集成方法（比如使用 CWM）可以有效降低元数据集成的成本和

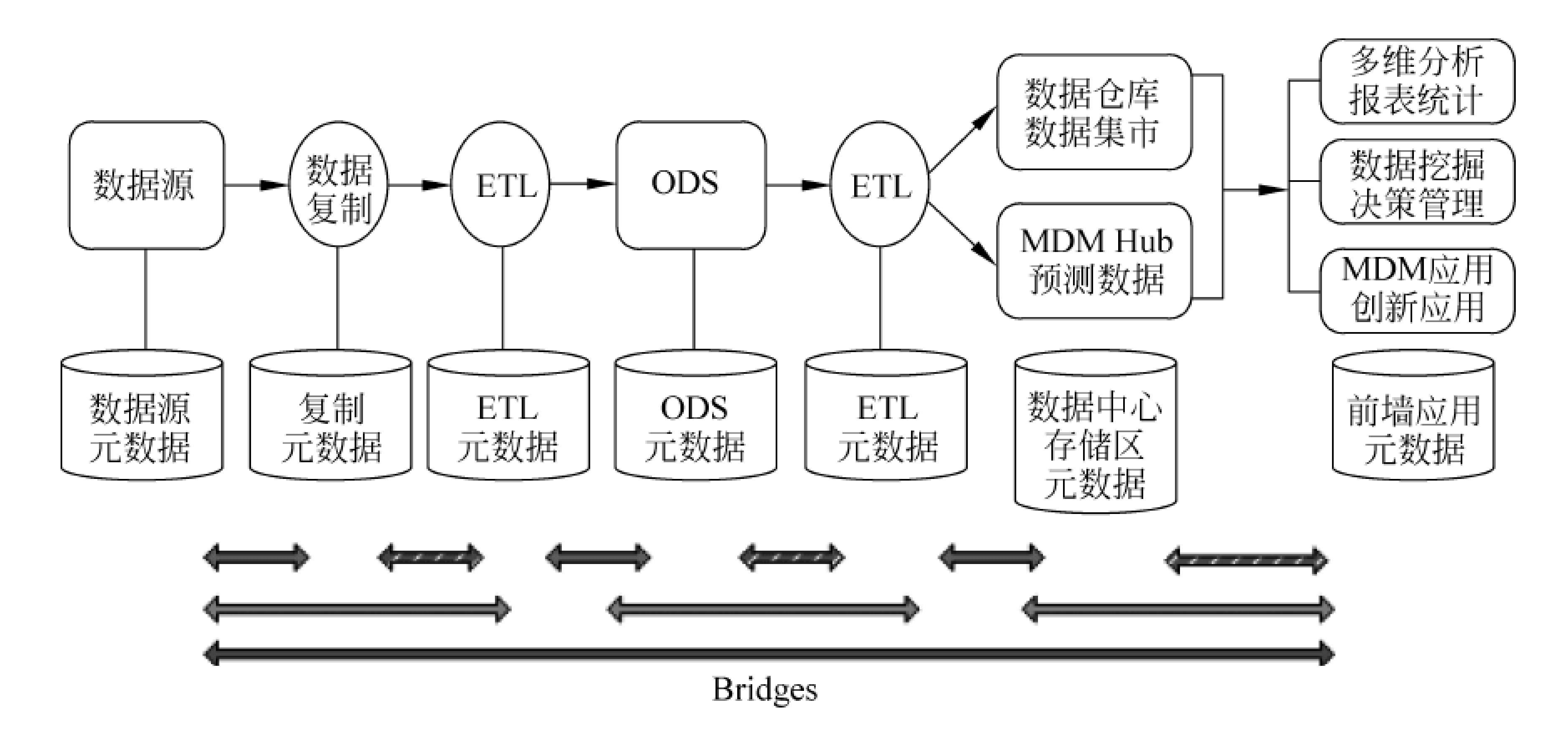


图 4.34　点对点的元数据集成体系结构

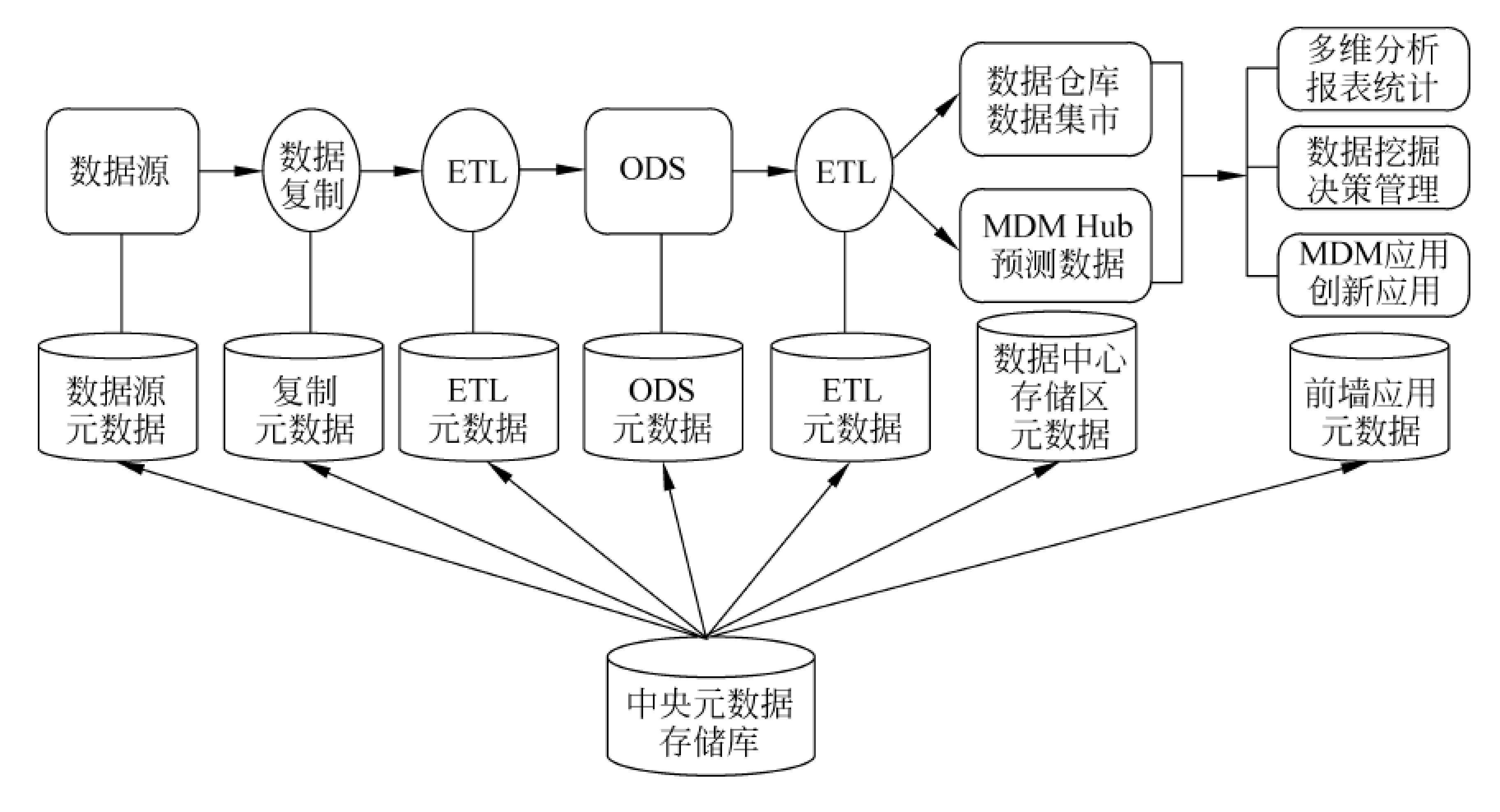


图 4.35　中央辐射式元数据体系结构

复杂度，无论点对点元数据集成体系结构还是中央辐射式元数据集成体系结构都可以因此受益。在点对点体系结构中，通过使用基于模型的方法可以不必在每一对需要集成的产品之间构建元数据桥，每个产品只需要提供一个适配器(Adapter)即可实现各个产品之间的元数据交换，适配器既了解公共的元模型也了解本产品元模型的内部实现。如图 4.36 所示，基于 CWM 模型驱动点对点元数据集成体系结构使用通用元模型，不再需要在各个产品间建立元数据桥，在各个产品之间通过适配器实现了语义等价性。

如图 4.37 所示，在基于模型驱动(比如 CWM)的中央辐射式元数据体系结构中，中央存储库包含公共元模型和整个领域(Domain)用到的该元模型的各个实例(模型)、存储库自身元模型及其实例、理解元模型(公共元模型和自身元模型)的适配器层，当然存储库也可以直接实现公共元模型的某些内部表示。

如图 4.38 所示，这种体系架构是基于 CWM 模型驱动的中央存储库元数据集成体系结构的一个变种，两个中央辐射式的拓扑结构通过各自的元数据存储库连接起来，也被称为分布式(Distributed)或联邦(Federated)体系结构。两个元数据存储库之间通过元数据桥连

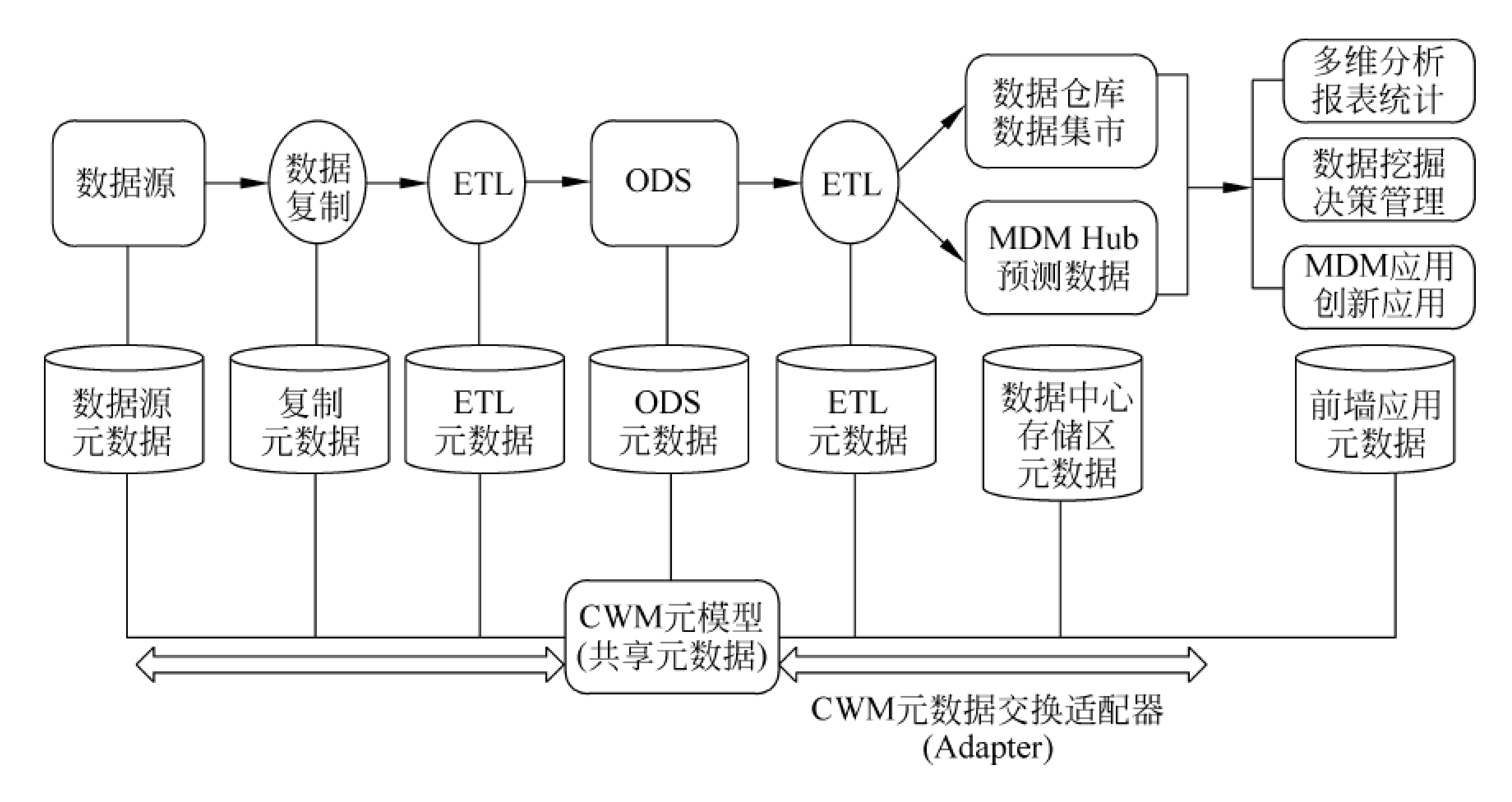


图 4.36　基于 CWM 模型驱动的点对点元数据集成体系结构

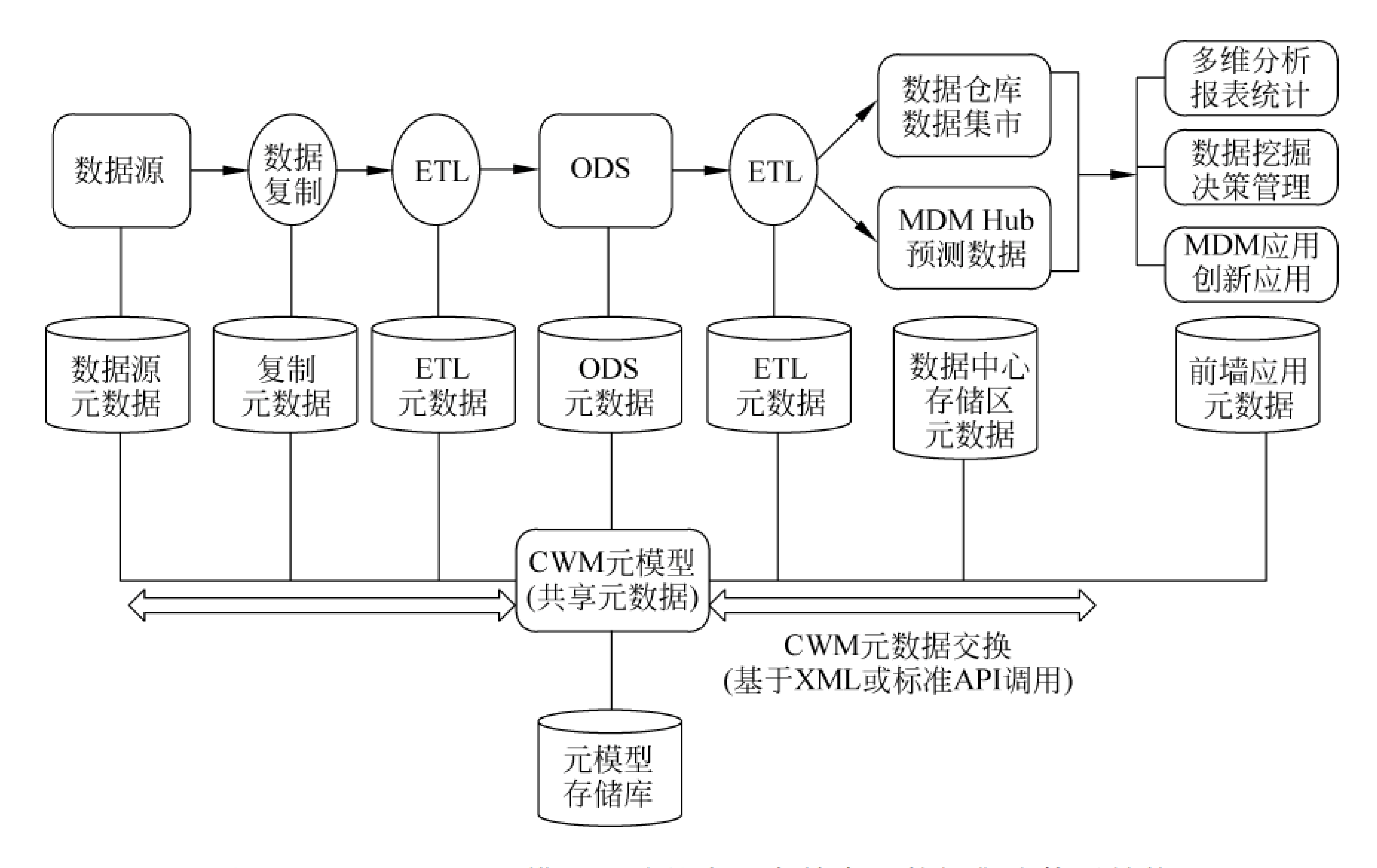


图 4.37　基于 CWM 模型驱动的中央存储库元数据集成体系结构

接，两个存储库可以使用相同的元模型和接口，也可以使用不同的元模型和接口。建立分布式元数据集成体系结构的原因有很多种，比如企业基于多个区域单独部署自己的应用，每个区域有自己的数据中心。

如图 4.39 所示，这种体系结构是分布式体系结构的变体，根存储库实现了元模型的公共部分(横跨整个企业)，叶子存储库实现了一个或多个特定的公共元模型子集，并只保存这些自己所对应的元数据实例。特定客户可以主要访问其感兴趣的元数据所在的叶子存储库，也可以访问其他叶子存储库和根存储库。这种体系结构被称为层次或星型拓扑结构。

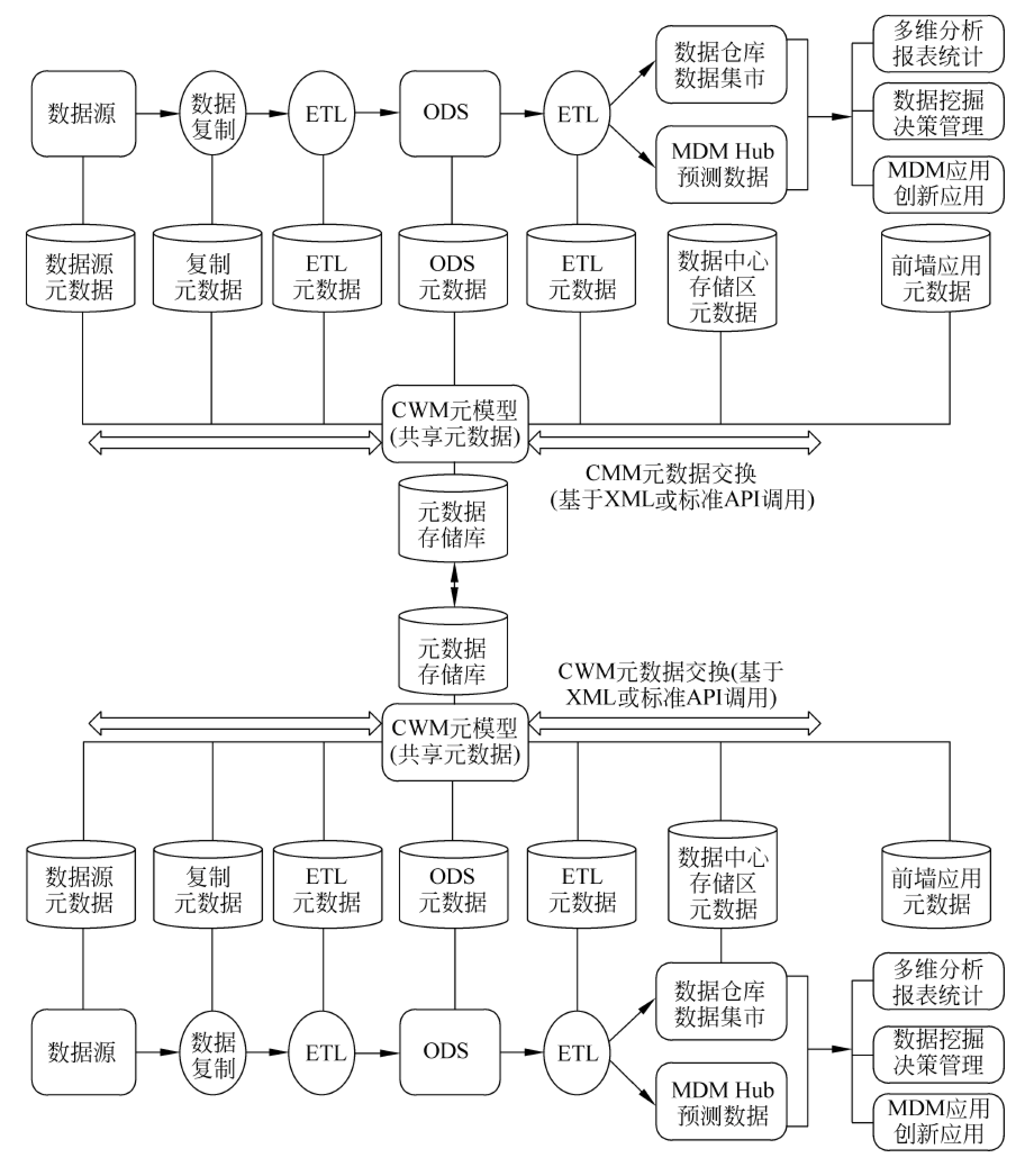


图 4.38 分布式(联邦式)元数据集成体系结构

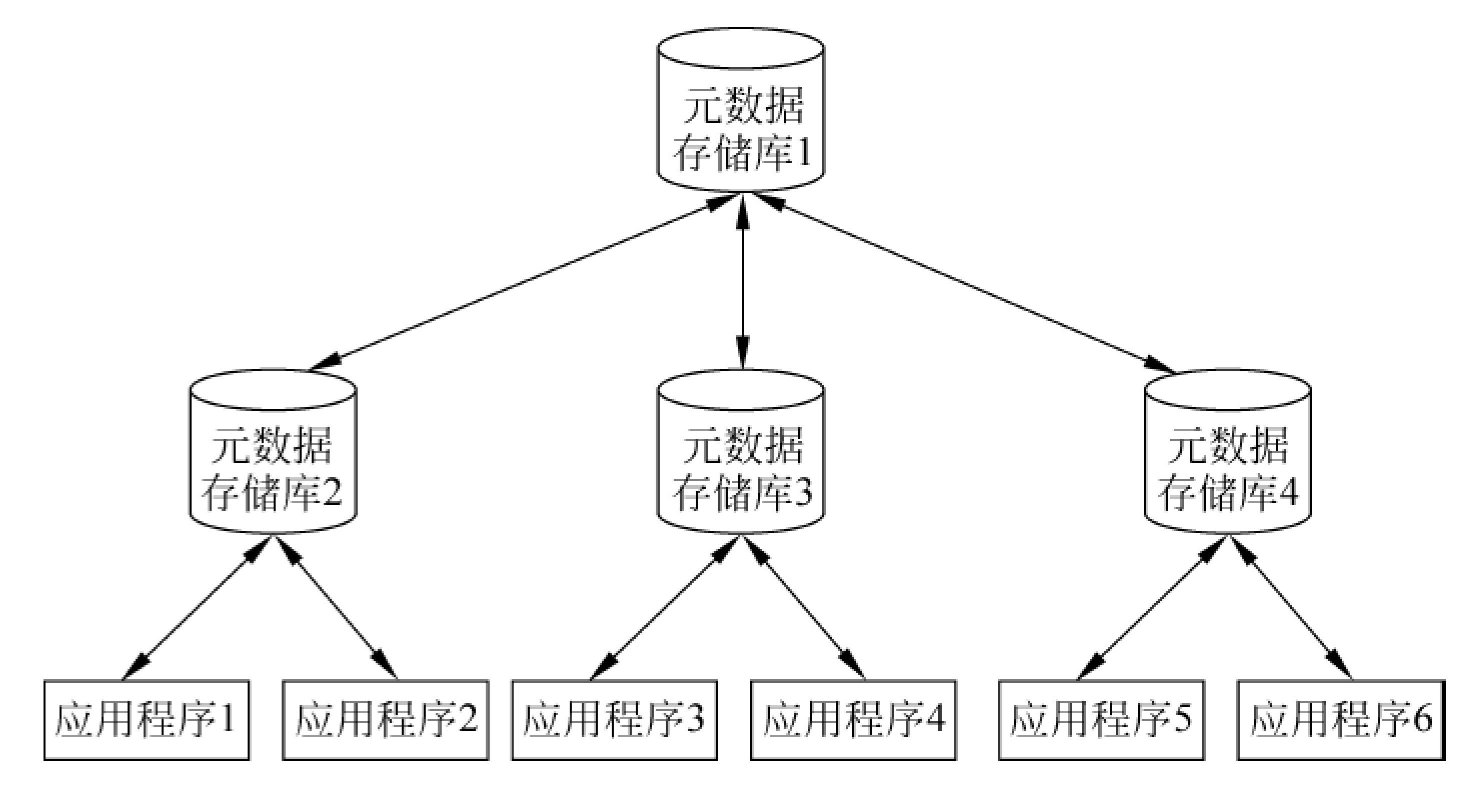


图 4.39 层次或星型元数据集成体系结构

4.5.3 实施元数据管理

在明确了元数据管理策略和元数据集成体系结构之后，企业可以根据需要选择合适的业务元数据和技术元数据管理工具，并制定相应的元数据管理制度进行全面的元数据管理。比如可以使用 IBM InfoSphere Business Glossary 进行业务元数据的管理，使用 IBM InfoSphere Metadata Workbench 作为元数据管理统一工具并进行图形化的元数据分析。

大数据扩大了数据的容量、速度和多样性，给元数据管理带来了新的挑战。在构建关系型数据仓库、动态数据仓库和关系型数据中心时进行元数据管理，有助于保证数据被正确地使用、重用并满足各种规定。同样，对大数据来说，元数据管理过程中出现的任何错误，都会导致数据重复、数据质量差和无法访问关键信息等问题。随着大数据技术在企业中的应用越来越广泛，企业需要在原有的元数据管理策略中增加大数据相关的内容。通常，大数据分析是受用例驱动的，企业可以通过梳理大数据用例的方式逐步完善大数据的元数据管理。

针对大数据的业务元数据，依旧可以通过构建基础本体、领域本体、任务本体和应用本体等方式来实现。通过构建基础本体，实现对级别且通用的概念的描述以及概念之间关系进行描述；通过构建领域本体，实现对于领域的定义，并确定该领域内共同认可的词汇、词汇业务含义和对应的信息资产等，提供对该领域知识的共同理解；通过构建任务本体，实现任务元素及其之间关系的规范说明或详细说明；通过构建应用本体，实现对特定应用的概念描述，其是依赖于特定领域和任务的。这样就通过构建各种本体，在整个企业范围提供一个完整的共享词汇表，保证每个元数据元素在信息供应链中每个组件的语义上保持一致，实现语义等效。

为了实现信息供应链中各个组件元数据的交互和集成，大数据平台的元数据集成体系结构依然可以采用基于模型驱动的中央辐射式元数据体系结构。对大数据平台中的结构化数据的元数据管理可以遵循公共仓库元模型（CWM）构建元数据体系结构，以便实现各个组件间元数据的交互；对大数据平台中的半结构化和非结构化数据的元数据管理，因为业内还没有通用的公共元模型，企业可以尝试采用基于自定义模型驱动的方式构建中央辐射式元数据体系结构。

简单来说，企业可以尝试以下步骤进行大数据的元数据管理：

（1）考虑到企业可以获取数据的容量和多样性，应该创建一个体现关键大数据业务术语的业务定义词库（本体），该业务定义词库不仅仅包含结构化数据，还可以将半结构化和非结构化数据纳入其中。

（2）及时跟进和理解各种大数据技术中的元数据，提供对其连续、及时的支持，比如 MPP 数据库、流计算引擎、Apache Hadoop/企业级 Hadoop、NoSQL 数据库以及各种数据治理工具如审计/安全工具、信息生命周期管理工具等。

（3）对业务术语中的敏感大数据进行标记和分类，并执行相应的大数据隐私政策。

（4）将业务元数据和技术元数据进行链接，可以通过操作元数据（如流计算或 ETL 工具所生成的数据）监测大数据的流动；可以通过数据世系分析（血缘分析）在整个信息供应链中实现数据的正向追溯或逆向追溯，了解数据都经历了哪些变化，查看字段在信息供应链各组件间转换是否正确等；可以通过影响分析了解具体某个字段的变更会对信息供应链中其他组件的字段造成哪些影响等。

(5) 扩展企业现有的元数据管理角色,以适应大数据治理的需要,比如可以扩充数据治理管理者、元数据管理者、数据主管、数据架构师以及数据科学家的职责,加入大数据治理的相关内容。

在实施元数据管理的过程中,可以参照元数据管理的成熟度模型确定企业当前元数据管理所在层次,并根据业务需要制定路线图实现元数据管理水平的提升。元数据管理成熟度模型具体如图 4.40 所示。

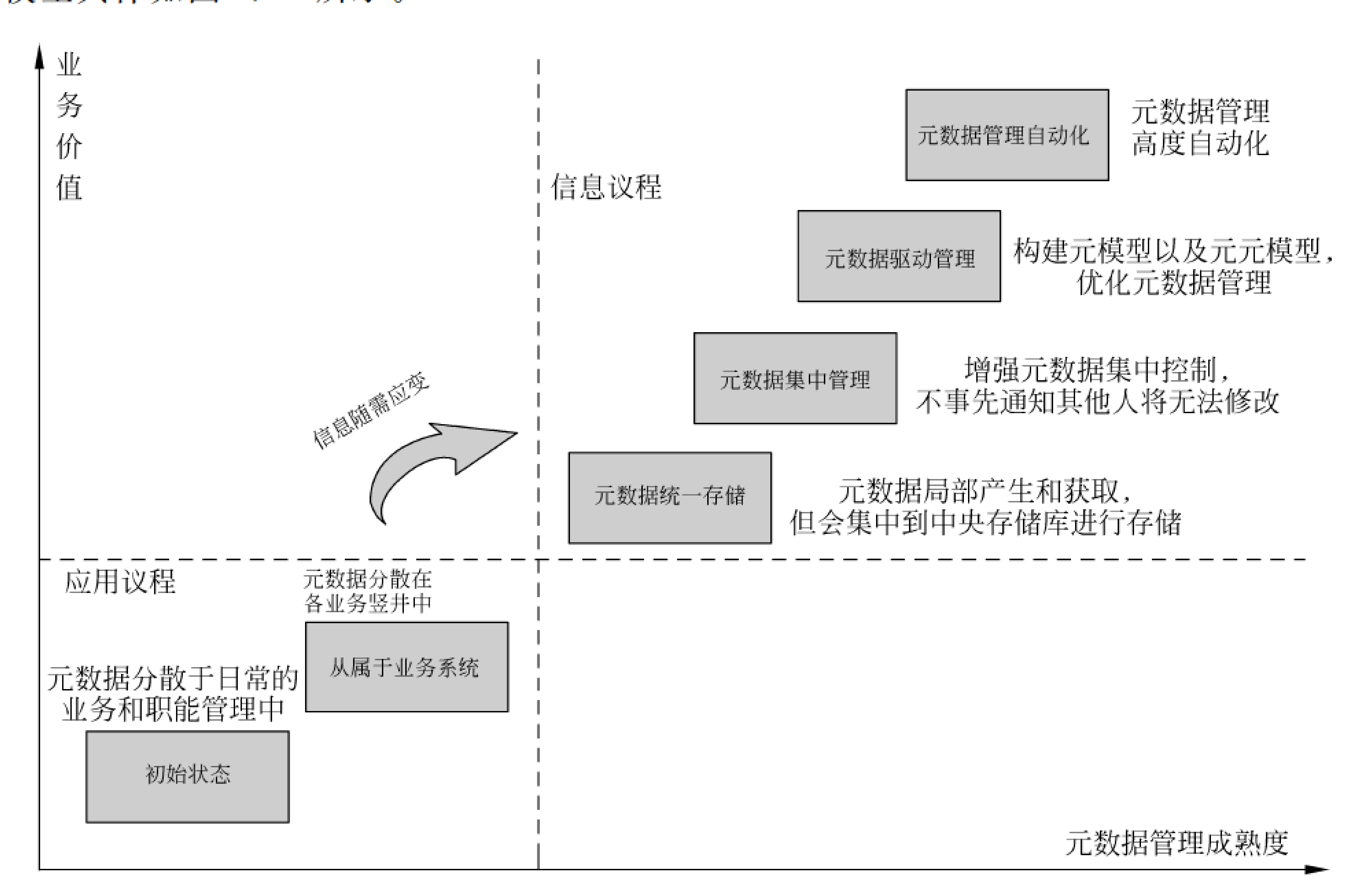


图 4.40 元数据管理成熟度模型

根据元数据管理的成熟度,大体可以分成 6 个级别。

(1) L0:初始状态

元数据分散于日常的业务和职能管理中,由某个人或某一组人员在局部产生或获取,并在局部使用,其他人如果想获得该元数据需要找到相应的人进行沟通获取。

(2) L1:从属于业务系统

在这个阶段,随着各个业务系统自动化构建完成,相应的元数据也随着需求整理、设计、开发、实施和维护等过程被各个业务系统孤立地全部或部分管理起来。业务元数据可能分散在各种业务规章、流程规定、需求、需求分析和概要设计等文档以及业务系统中,技术元数据可能分散在详细设计、模型设计和部署方案等各种文档和各种中间件以及业务系统中。由于各个业务系统处于一个个竖井之中,元数据之间互通互联困难,如果需要获取其他系统的元数据,除了调阅各种文档外,对分散在各种中间件和业务系统中的技术元数据需要通过桥(Bridge)的方式实现互通互联。

(3) L2:元数据统一存储

元数据依然在局部产生和获取,但会集中到中央存储库进行存储,业务元数据会手工录入到中央存储库中,技术元数据分散在文档中的部分也通过手工录入到中央存储库中,而散

落在各个中间件和业务系统中的技术元数据则通过桥(Bridge)的方式被读取到中央存储库中。业务元数据和技术元数据之间全部或部分通过手工方式做了关联。中央存储库的构建,使得元数据在整个企业层面可被感知和搜索,极大地方便了企业获取和查找元数据。缺点是,元数据仍然在各业务系统上维护,然后更新到中央存储库,各业务竖井之间仍然使用不同的命名法,经常会造成相同的名字代表不同意义的事情,而同一件事情则使用了多个不同的名字,有些没有纳入业务系统管理的元数据则容易缺失。元数据没有有效的权限管理,局部元数据更改后也不自动通知其他人。

(4) L3:元数据集中管理

L3 在 L2 的基础上做了改进,增强了元数据的集中控制,局部业务单元或开发小组如不事先通知其他人,将无法对元数据进行修改。局部元数据的修改完成后将会广播给其他人。和其他中间件和应用系统的交互,仍然通过桥接的方式进行,中央存储库中的业务元数据和技术元数据之间还是通过手工方式进行映射。

(5) L4:元模型驱动管理

在 L3 的基础上,通过构建元模型以及元元模型,优化各业务单元之间的各种冲突和各种副本,创建、管理和共享业务词汇表和分类系统(基于主题领域的层次结构)。业务词汇表(业务元数据)包含与企业相关的词汇、词汇业务含义以及词汇与信息资产(技术元数据)的关系,可以有效帮助企业用户了解其业务元数据和技术元数据对应的业务含义。分类是基于主题领域的层次结构,用以对业务术语归类。和其他中间件和应用系统的交换,通过基于 CWM 的适配器方式进行连接。

(6) L5:元数据管理自动化

在 L5 元数据管理是高度自动化的,当逻辑层次元数据变更时,会被传播到物理层次,同样物理层次变更时逻辑层次将被更新。元数据中的任何变化将触发业务工作流,以便其他业务系统进行相应地修改。由于各个业务系统遵照相同的业务词汇表和分类系统(元模型),它们之间的关系可以通过知识本体进行推断,因此各个应用系统之间的数据格式的映射自动产生。

4.6 数据的交付

大数据涵盖了众多社会生活的范畴,从金融交易到人类基因组,从汽车的遥测传感器到互联网上社会媒体日志。利用传统的数据库方式来处理和存储这些大数据是相当昂贵的。为了解决这个问题的新技术,利用开放源解决方案和商业硬件高效存储数据,并行工作负载,提供快速处理能力。

事实上,大家所看到的新出现的大数据需求,以及关于其处理平台和流程的讨论只是大数据传输整体的一部分。在现实中,实现全部潜在大数据的交付过程,需要 7 个步骤:

- 收集:从数据源和分布在多个节点处收集数据,通常是一个网格、每个进程的一个子集的并行数据。
- 流程:对每个节点上的数据进行快速计算。节点"压缩"结果数据到更多的消费数据,由此产生的数据集可以被人工(在分析的情况下)或机器(在解释大型结果的情况下)使用。

- 管理：正在处理的大数据往往是异构的，来自不同的交易系统。这些数据通常需要理解、定义、注释，并且安全起见，还要进行扫描和审核。
- 测量：公司往往会测量数据的速率，可与其他客户的行为或记录进行整合，并随着时间的推移来决定是否对其进行整合或校正。业务要求应告知测量和持续跟踪的类型。
- 消耗：所产生的使用数据应符合原要求的处理流程。例如，如果分析几百 TB 的社会媒体数据，有助于了解社会媒体数据如何驱动用户额外购买产品，那么就应该建立如何访问和更新社会媒体数据的规则。这与机器对机器的数据访问是同样重要的。
- 存储：由于“数据即服务”趋势的形成，越来越多的数据开始存储在单一位置，以便于进程的访问。数据用于短期存储还是长期保留，应有谨慎的存储方案。
- 数据治理：数据治理适用于前六个阶段。通过建立流程和指导原则，制裁围绕数据的行为。大数据需要根据其预期消费进行管辖。

大多数工作人员负责调查和获取大数据解决方案时侧重于收集和存储步骤，而忽略了其他的步骤。他们的问题是：“我们如何收集所有这些数据，我们把这些数据存储在何处？”

正如 *ITBusinessEdge* 一书的作者洛林劳森在其最近的博客中所说，“确保您的分享是合理性的唯一途径是要保证您有一套有效的管理大数据的计划。”

挖掘数据治理进程，尽最大的努力确保数据：

- 商业价值和理想的结果是明确的；
- 处理关键数据的相关政策已经被批准；
- 专业知识应用到大数据问题；
- 定义关键数据的规则是明确的；
- 冲突和问题升级有一个过程；
- 数据管理；
- 在关键问题发展阶段有决策权；
- 执行数据隐私政策。

总之，数据治理意味着大数据的应用程序是有用的和相关的。确保不会浪费新的大数据，使得处理、存储和交付速度更具成本效益，比以往的技术更灵活。



IBM InfoSphere软件

5.1 InfoSphere Data Architect

IBM InfoSphere Data Architect 是一个企业级的数据建模工具，它拥有一套完整的开发环境能够帮助用户对异构分布的数据资产进行建模，并且发现和创建这些数据资产之间的关系。

通过本章学习可以了解到：

- 什么是 IBM InfoSphere Data Architect；
- 如何使用 IBM InfoSphere Data Architect；
- 系统配置要求；
- 如何安装 IBM InfoSphere Data Architect。

5.1.1 什么是 IBM InfoSphere Data Architect

当前企业拥有数量庞大的数据，并且许多决策的制定都依赖于这些数据。IT 业面临非常明确的挑战：了解数据、提高数据的质量和确保数据的一致性、确保数据的设计满足商业需求。IBM InfoSphere Data Architect 是一个协作的数据建模和信息集成工具，用于发现、建模、关联和标准化各种数据资产。它的主要功能和优点包括：

- InfoSphere Data Architect 支持对多种数据源进行分析并发现其数据结构。通过 JDBC 和数据源建立连接后，用户使用 InfoSphere Data Architect 可以方便地浏览数据库中的数据结构，它还提供了易于操作和理解的界面，对数据库中的对象按照层次结构分类，便于用户对数据库中的每一个对象的详细信息进行查看和理解。
- InfoSphere Data Architect 支持多种数据库，如 DB2®、Informix®、Oracle、Sybase、Microsoft SQL Server、MySQL 和 Teradata 等。用户可以为这些数据库创建逻辑数据模型、物理数据模型和域模型。逻辑数据模型和物理数据模型中的元素还可以用图进行可视化展现。图的展现采用信息工程(IE)表示法显示，同时，在物理数据模型中的图还可以通过 UML 表示法显示。对于物理数据模型的创建，InfoSphere

Data Architect 不但支持从头开始设计创建物理数据模型，还支持从逻辑数据模型转换为物理数据模型和通过反向设计创建物理数据模型。

IBM InfoSphere Data Architect 还支持创建和设计多维逻辑数据模型和多维物理数据模型。同样地，用户可以从无到有地创建新的多维模型，也可以通过反向设计从已有的数据源中创建。

- 大多数数据开发项目都采用迭代开发的方式。因此，模型设计中的增量变化控制管理和减少增量变化带来的影响非常重要。InfoSphere Data Architect 可以帮助解决这个问题。分析影响功能会帮助列出所有与选中数据对象相关的数据对象。比较和同步功能则可以对两个模型、模型和数据库，或者两个数据库进行对比分析，并且支持用户在模型或目标数据库中进行设计的改变。用户还可以使用 Optim™ Database Administrator 来进行复杂的 DB2 数据库对象的改变和数据迁移。
- InfoSphere Data Architect 支持用户设计和实现统一的标准，这些统一的标准可以用来提升企业数据的质量、规范数据在命名、含义、取值、关联、权限、隐私以及可跟踪性方面的一致性。这些标准一旦定义好，可以应用于不同种类的模型和数据库。同时，InfoSphere Data Architect 还提供了内置的、可扩展的、基于规则的分析验证功能，用来对数据模型或者数据库进行命名、语法、范式化以及是否满足最佳实践等方面的验证。
- 不论是对于小的工作团队中每个人承担多种职能的情况，或者是对于一个大的团队里每个人有明确的工作划分的情况，都可以将 InfoSphere Data Architect 与基于 Eclipse 的版本控制软件集成起来使用，如 Rational® Clear Case 或者并发版本控制系统（CVS）。

5.1.2 下载 DB2 Express-C

读者可以从 IBM Academic Initiative 处索取到免费版本的 IBM InfoSphere Data Architect。但是读者必须隶属于某一个大学并且满足下面的条件之一：

- 数据建模研究方向的学生
- 数据建模或数据架构方向的讲师
- 使用本产品来辅助学术研究

5.1.3 安装 InfoSphere Data Architect

读者可以使用 Launchpad 界面方式手动安装或者使用配置文件方式自动安装 IBM InfoSphere Data Architect。使用配置文件自动安装指的是可以将安装需要的选项写在配置文件中，然后运行配置文件进行安装，安装过程不需要用户干涉，这种方式在多台机器上需要进行安装时非常有用。

在这节里将重点介绍使用 Launchpad 界面的安装方式。IBM InfoSphere Data Architect 的安装需要在 IBM Installation Manager 中进行，下面假设读者还没有安装 IBM Installation Manager，如果已经安装了相同版本的 IBM Installation Manager，就不需要再次安装。

安装之前需要拥有目标机器操作系统的管理员权限。请按照以下步骤进行 IBM

InfoSphere Data Architect 产品的安装：

(1) 解压文件，然后运行 launchpad。

- Windows 操作系统：在解压后的文件夹中运行 setup.exe 文件。
- Linux 或 UNIX 操作系统：在解压后的根目录中运行 setup 命令。运行后会打开 launchpad。

(2) 选择左侧列表中的“安装产品”。在安装产品页面，可以选择需要的安装方式，如图 5.1 所示。



图 5.1　选择安装产品运行 Installation Manager

选择【管理安装】命令，IBM Installation Manager 将会被打开，如图 5.2 所示。



图 5.2　选择要安装的软件包

为共享资源和 Installation Manager 选择安装目录。在练习中，可以使用默认的路径，但是需要注意确保目标磁盘有足够的空间以备之后可以和其他基于 Eclipse 的产品进行 shell-share。

注意：如果之后使用 Installation Manager 安装产品，将不能选择共享资源的目录。

单击【下一步】按钮，如果在目标机器上已经有安装好的 Eclipse 并且版本号为 3.4.2，可以选择扩展已有的 Eclipse 开发环境，使用这种方式将只安装产品特有的功能，不会改变已有的 Eclipse 开发环境的设置。对于练习，这里使用默认设置，单击【下一步】按钮，如图 5.3～图 5.10 所示。



图 5.3　接受许可协议



图 5.4　选择安装目录



图 5.5　创建新的软件包组

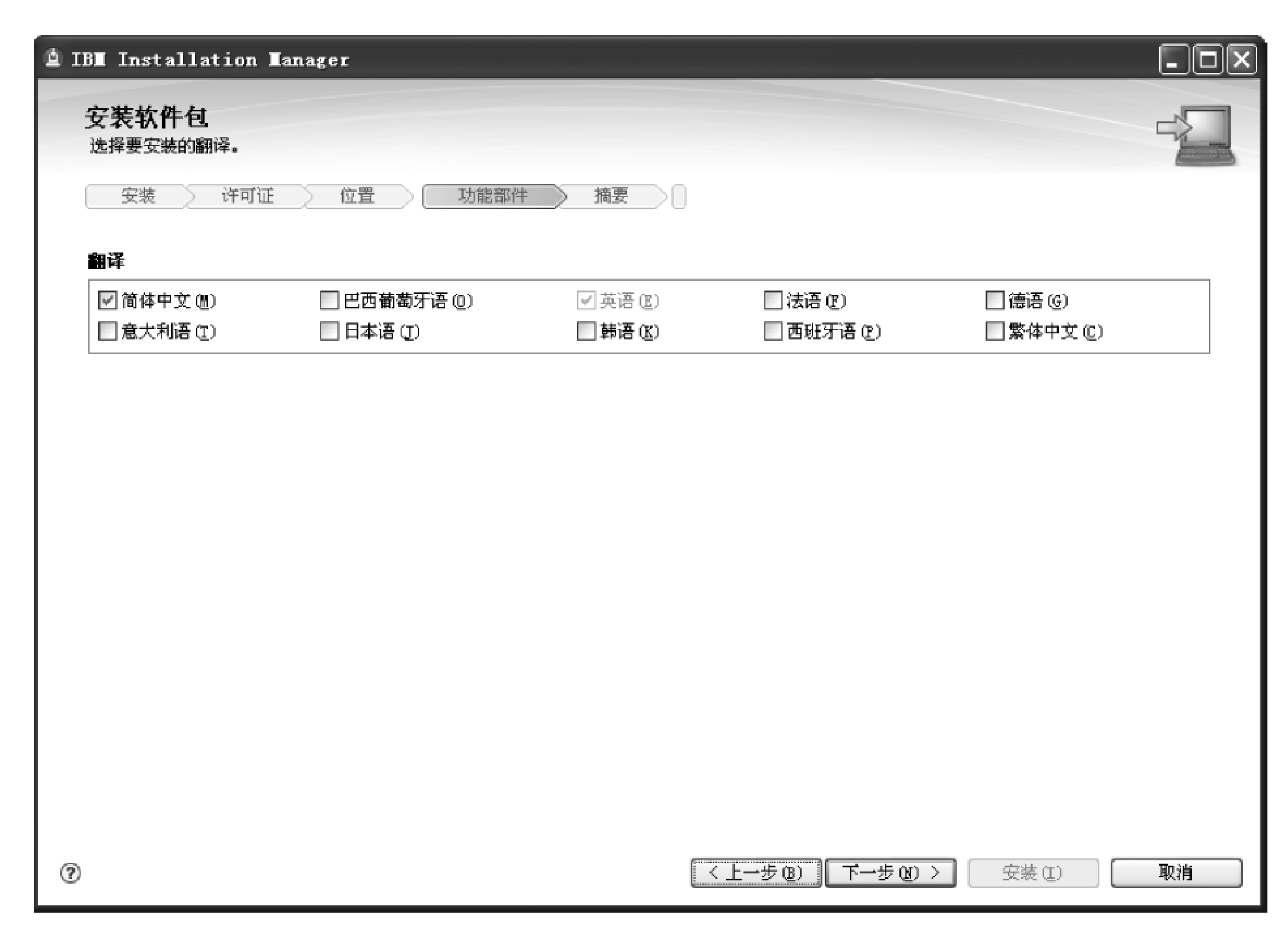


图 5.6　选择语言包



图 5.7 选择要安装的功能部件



图 5.8 选择从何处获取软件帮助

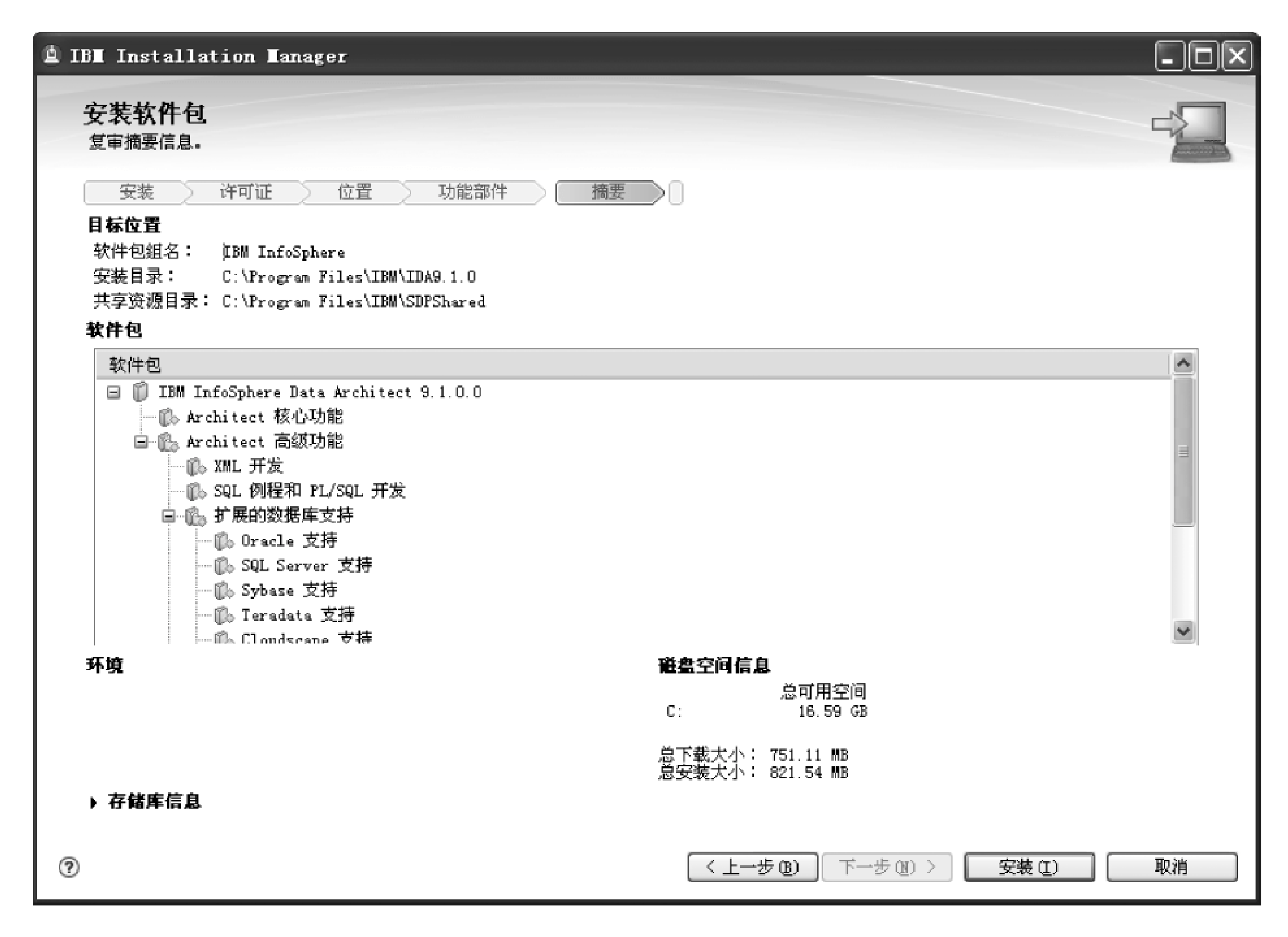


图 5.9 确定安装摘要，选择安装进行安装

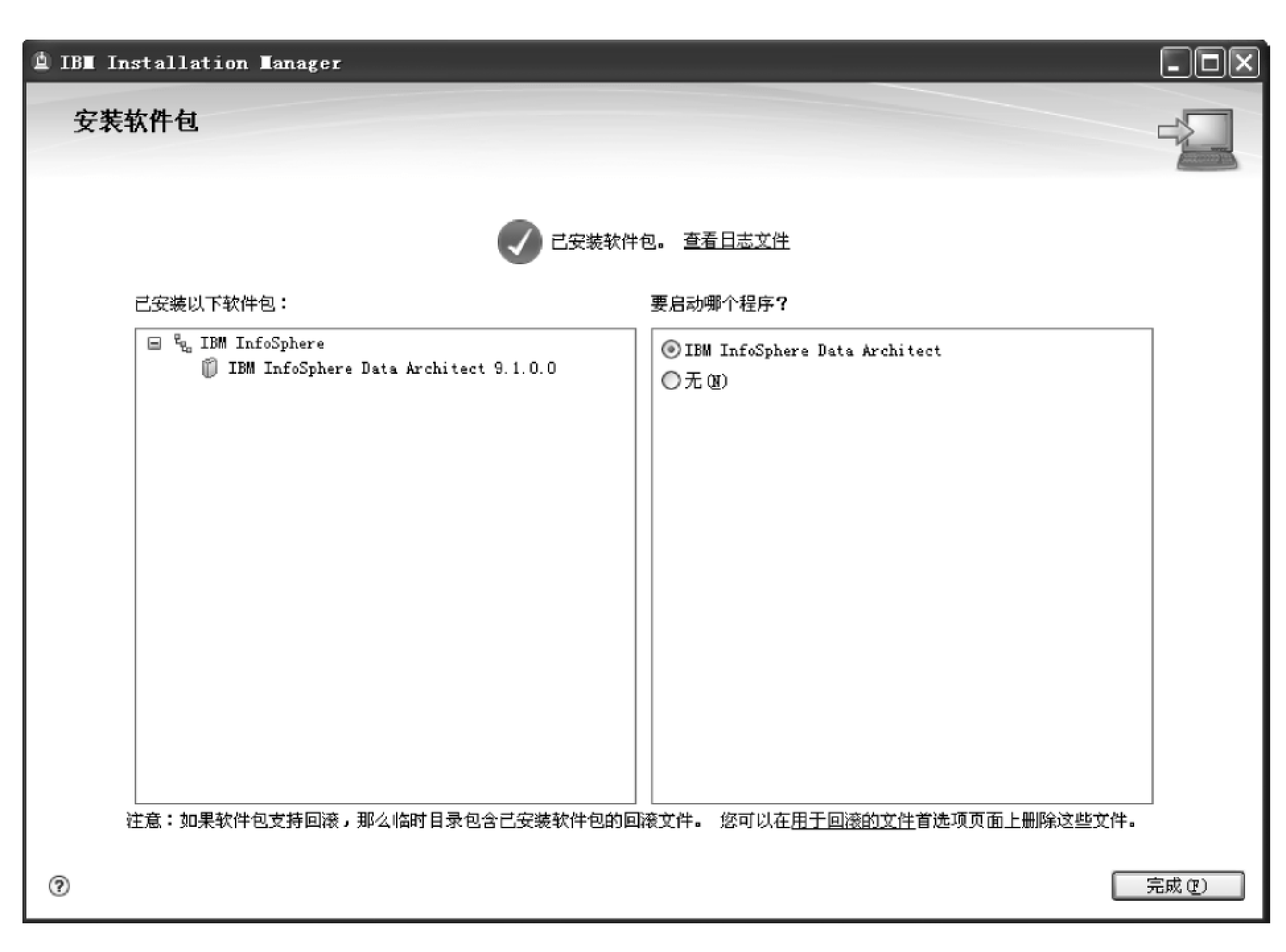


图 5.10 产品安装成功界面

在图 5.10 所示的页面中，选择【无】选项，先不启动 InfoSphere Data Architect。如果读者购买了产品的许可证，在启动 InfoSphere Data Architect 之前，需要首先使用 Installation Manager 为产品应用许可证。单击【完成】按钮退出安装界面。

在 launchpad 左侧选择【退出】命令，退出 launchpad。

到此已经成功地安装了 IBM InfoSphere Data Architect。如果读者拥有产品的许可证，需要在试用版 30 天有效期之前应用许可证激活产品。

5.2 InfoSphere Streams

来自多个来源的信息正在以难以置信的速度增长。互联网用户数量在 2012 年已经达到 22.7 亿。每一天，Twitter 都会生成超过 12TB 的 tweet，Facebook 生成超过 25TB 日志数据，纽约证券交易所采集近 1TB 交易信息。每天会创建大约 300 亿个射频识别(RFID)标记。此外，每年销售的数亿台 GPS 设备，目前正在使用的超过 3000 万个连网的传感器(而且每年在以高于 30%的速度增长)，都在产生数据。这些数据量预计在未来 10 年中每 2 年就会翻一番。

一家公司在一年时间内可生成高达数 PB 的信息：网页、博客、单击流、搜索索引、社交媒体论坛、即时消息、文本消息、电子邮件、文档、用户人口统计数据、来自主动和被动系统的传感器数据，等等。许多人估计，这些数据中高达 80%都是半结构化或非结构化数据。公司一直在寻求更加敏捷地经营业务，以更加创新的方式执行数据分析和决策流程。而且他们认识到，这些流程中损失的时间可能导致错失业务机会。大数据挑战的核心是，公司掌握轻松地分析和理解互联网级信息的能力，就像他们现在可分析和理解较少量结构化信息一样。

图 5.11 展示了从极大量、多样性和高速产生的上下文数据中提取洞察的大数据挑战，这在以前是无法做到的。

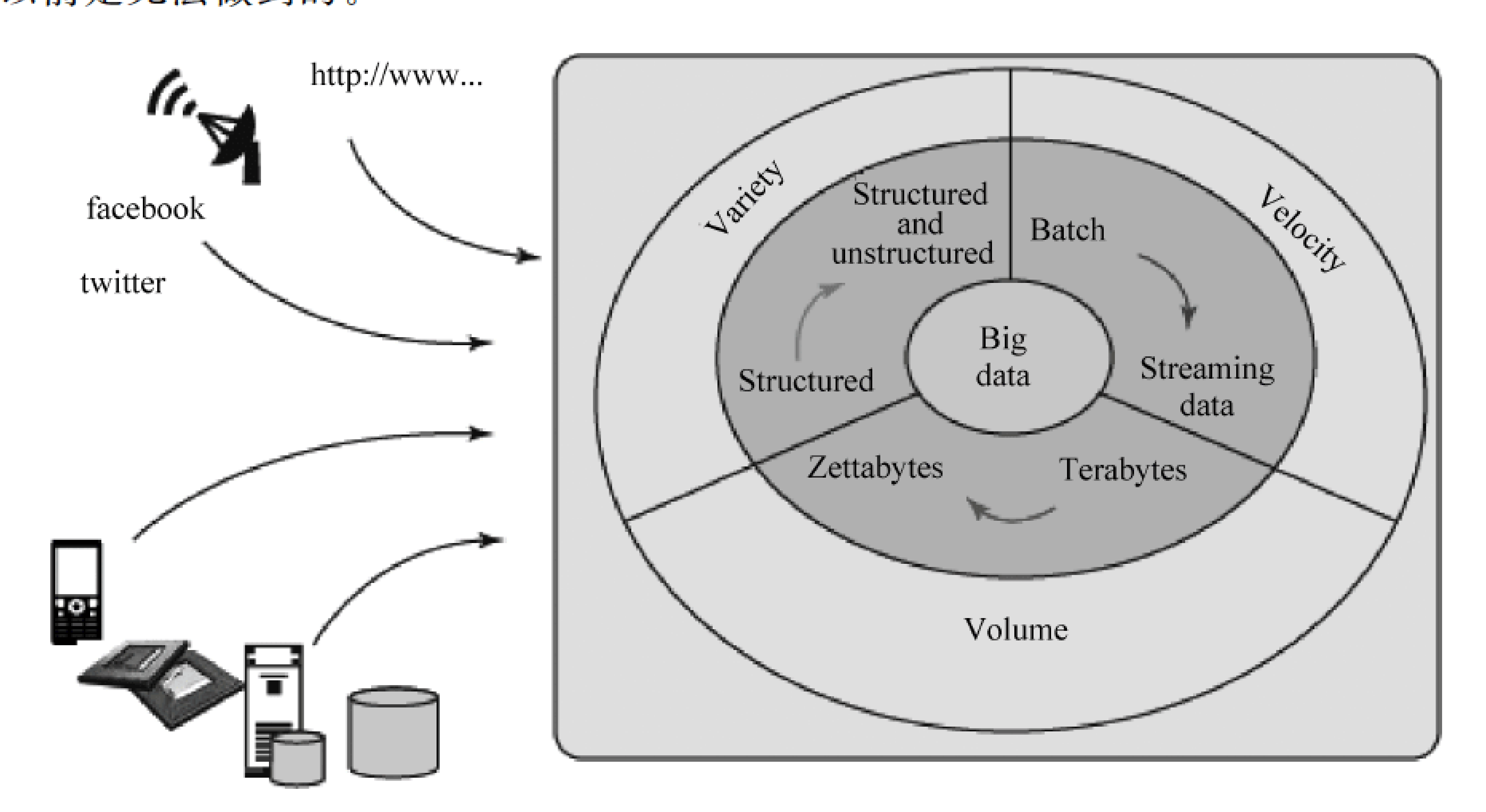


图 5.11 大数据挑战

在图 5.11 中，Volume 指数据规模，从 TB 级到 ZB 级。Variety 指许多不同结构中的数据复杂性，从关系数据到日志再到原始文本。Velocity 反映了流数据和大规模的数据移动。

IBM 正在帮助公司应对大数据挑战，为他们提供工具来集成和管理海量、高速产生的数据，应用原生格式的分析，可视化可用数据以进行专门分析，等等。下面将介绍 InfoSphere Streams，该技术支持同时分析许多数据类型并实时执行复杂计算。读者将了解 InfoSphere Streams 的工作原理，它的用途，以及如何结合使用它与另一个用于大数据分析的 IBM 产品(IBM InfoSphere BigInsights)来执行高度复杂的分析。

5.2.1 安装流计算：一种新的计算模式

流计算是新的数据声场场景所不可或缺的一种新计算模式，比如无处不在的移动设备、位置服务和遍布各处的传感器。人们需要可伸缩的计算平台和并行架构来处理生成的海量流数据。

BigInsights 技术不足以支持实时流处理任务，因为它们主要面向对静态数据的批处理的支持。在处理静态数据的过程中，列出所有已连网用户的查询会得到单一的结果集。借助对流数据的实时处理，可执行一种持续查询，比如列出在过去 10 分钟内连网的所有用户。此查询将返回持续更新的结果。在静态数据领域中，用户犹如在干草堆中捞针；在流数据领域中，用户可轻松地找到这根针，因为干草已被吹走。

图 5.12 演示了在静态数据上执行的计算与在流数据上执行的计算之间的区别。

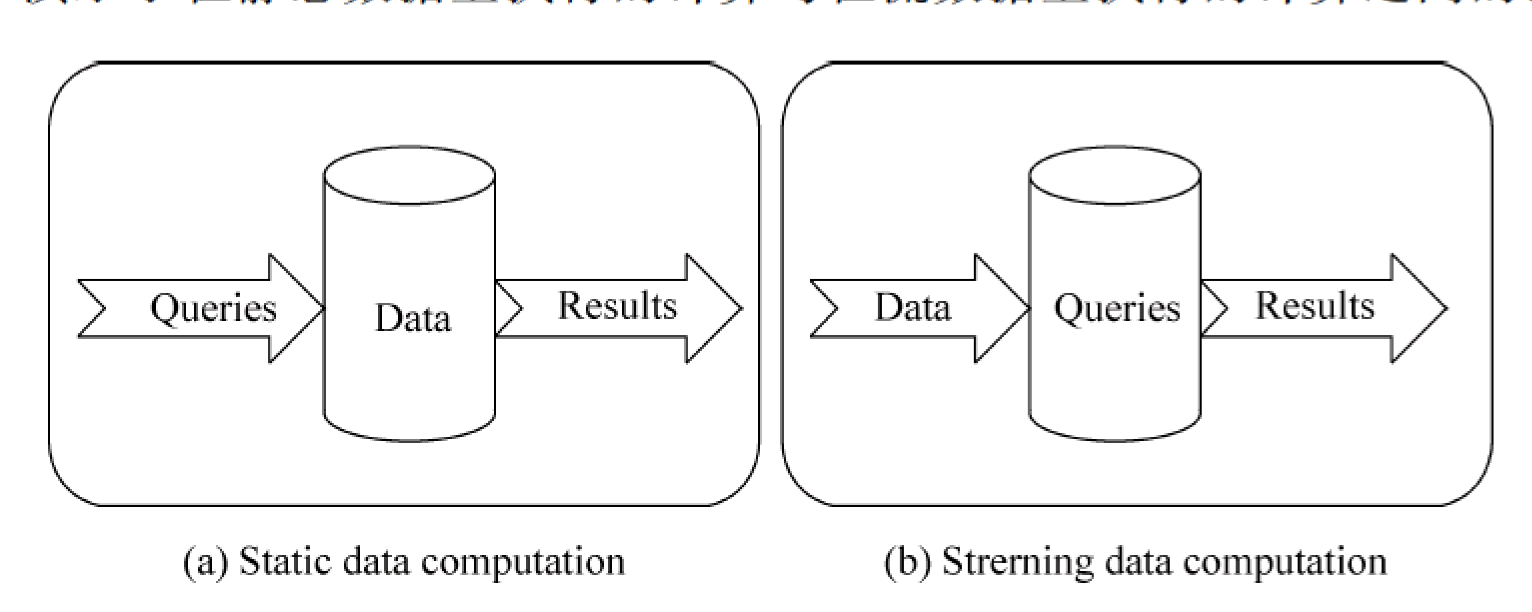


图 5.12 静态数据计算与流数据计算

在静态数据计算中(图 5.12(a))，查询有关静态数据的问题。在流数据计算中(图 5.12(b))，数据会由静态问题持续计算。

InfoSphere Streams 平台支持流数据的实时处理，支持不断更新持续查询的结果，可仍在移动的数据流中检测洞察。

5.2.2 InfoSphere Streams 概述

InfoSphere Streams 旨在从一个几分钟到几小时的窗口中的移动信息(数据流)中揭示有意义的模式。该平台能够获取低延迟洞察，并为注重时效的应用程序(比如欺诈检测或网络管理)获取更好的成果，从而提供业务价值。InfoSphere Streams 还可合并多个流，使得能够从多个流中获取新洞察，如图 5.13 所示。

InfoSphere Streams 的主要设计目的是：

- 快速响应事件和不断变化的业务条件与需求。

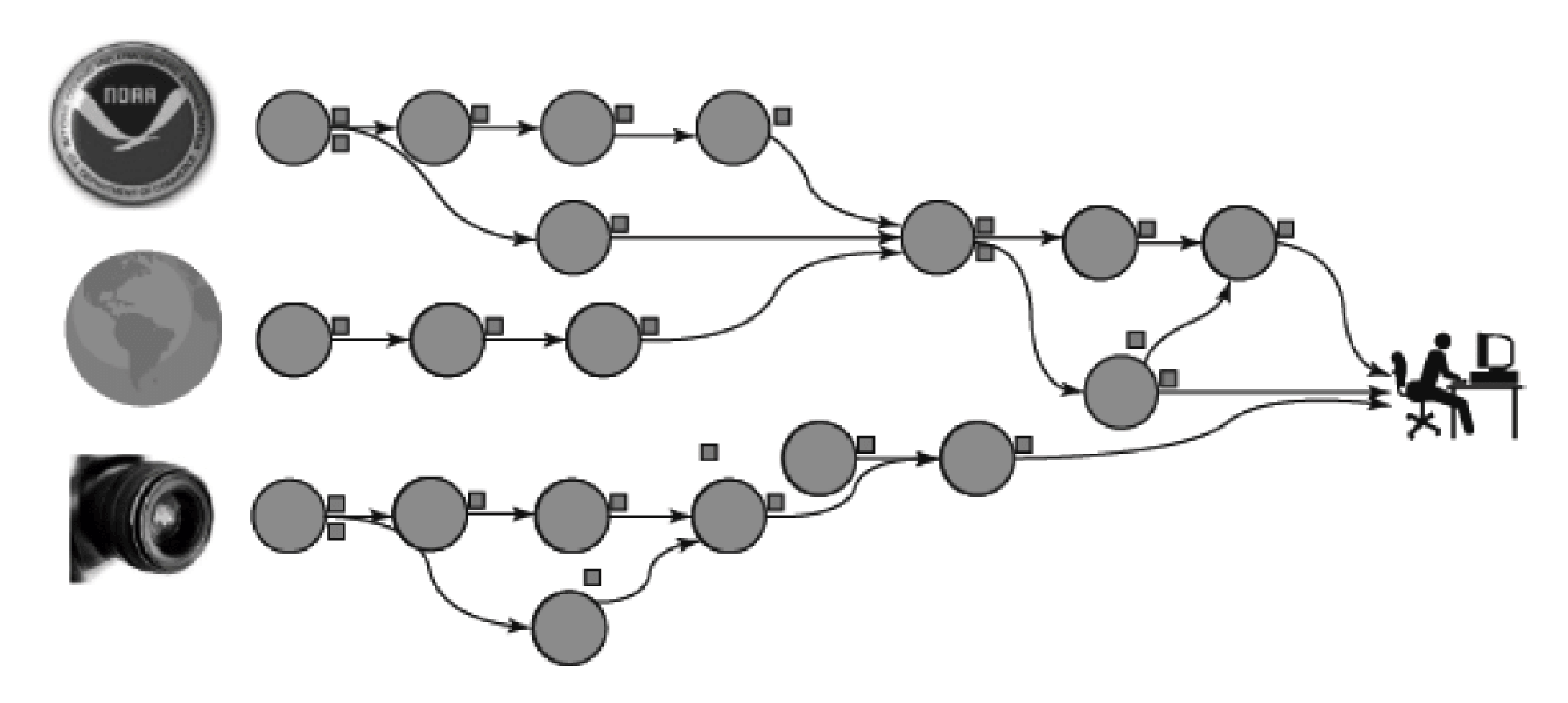


图 5.13 合并的流处理

- 支持以比现有系统快几个数量级的速度对数据执行持续分析。
- 快速适应不断变化的数据形式和类型。
- 管理新的流模式的高可用性、异构性和分布。
- 为共享的信息提供安全性和信息机密性。

InfoSphere Streams 提供了一种编程模型和 IDE 来定义数据来源，还提供了已融合到处理执行单元中的称为运算符的软件分析模块。另外还提供了基础架构来支持从这些组件合成可扩展的流处理应用程序。主要平台组件包括：

- 运行时环境：这包括平台服务，以及一个用于在单个主机或一组集成的主机上部署和监视 Streams 应用程序的调度程序。
- 编程模型：用户可使用 SPL（Streams Processing Language，流处理语言，一种声明性语言）来编写 Streams 应用程序。可使用该语言陈述用户的需求，运行时环境会承担确定如何最佳地服务该请求的责任。在此模型中，一个 Streams 应用程序表示为一个由运算符和连接它们的流组成的图表。
- 监视工具和管理接口：Streams 应用程序处理数据的速度比普通的操作系统监视实用程序快得多。InfoSphere Streams 提供了可处理此环境的工具。

5.2.3 流处理语言

SPL，InfoSphere Streams 的编程语言，是一种分布式数据流合成语言。它是一种类似 C++或 Java 的可扩展且全功能的语言，支持用户定义的数据类型。用户可以使用 SPL 或原生语言（C++或 Java）编写自定义函数。也可以使用 C++或 Java 编写用户定义的运算符。

InfoSphere Streams 持续应用程序会描述一个导向图，该图由各个互联且处理多个数据流的运算符组成。数据流可来自系统外部，或者在应用程序内部生成。SPL 程序的基本构建块包括：

- 流：一个无限的结构化元组序列。它可逐个元组地由运算符使用或通过一个窗口的定义来使用。
- 元组：属性及其类型的一个结构化列表。流上的每个元组拥有由其流类型指定的形式。

- 流类型：指定元组中每个属性的名称和数据类型。
- 窗口：一个有限、有序的元组分组。它可以基于计数、时间、属性值或标点符号。
- 运算符：SPL 的基础构建块，它的运算符会处理来自流的数据并可生成新流。
- 处理元素(PE)：基础执行单元。一个 PE 可封装单个运算符或多个合并的运算符。
- 作业：一个已部署好的用来执行的 Streams 应用程序。它由一个或多个 PE 组成。除了一组 PE 之外，SPL 编译器还会生成一个 ADL(Application Description Language，应用程序描述语言)文件来描述应用程序的结构。该 ADL 文件包含每个 PE 的详细信息，比如要加载和执行哪个二进制文件，调度限制、流格式和一个内部运算符数据流图。

图 5.14 演示了 SPL 程序的 InfoSphere Streams 运行时视图。

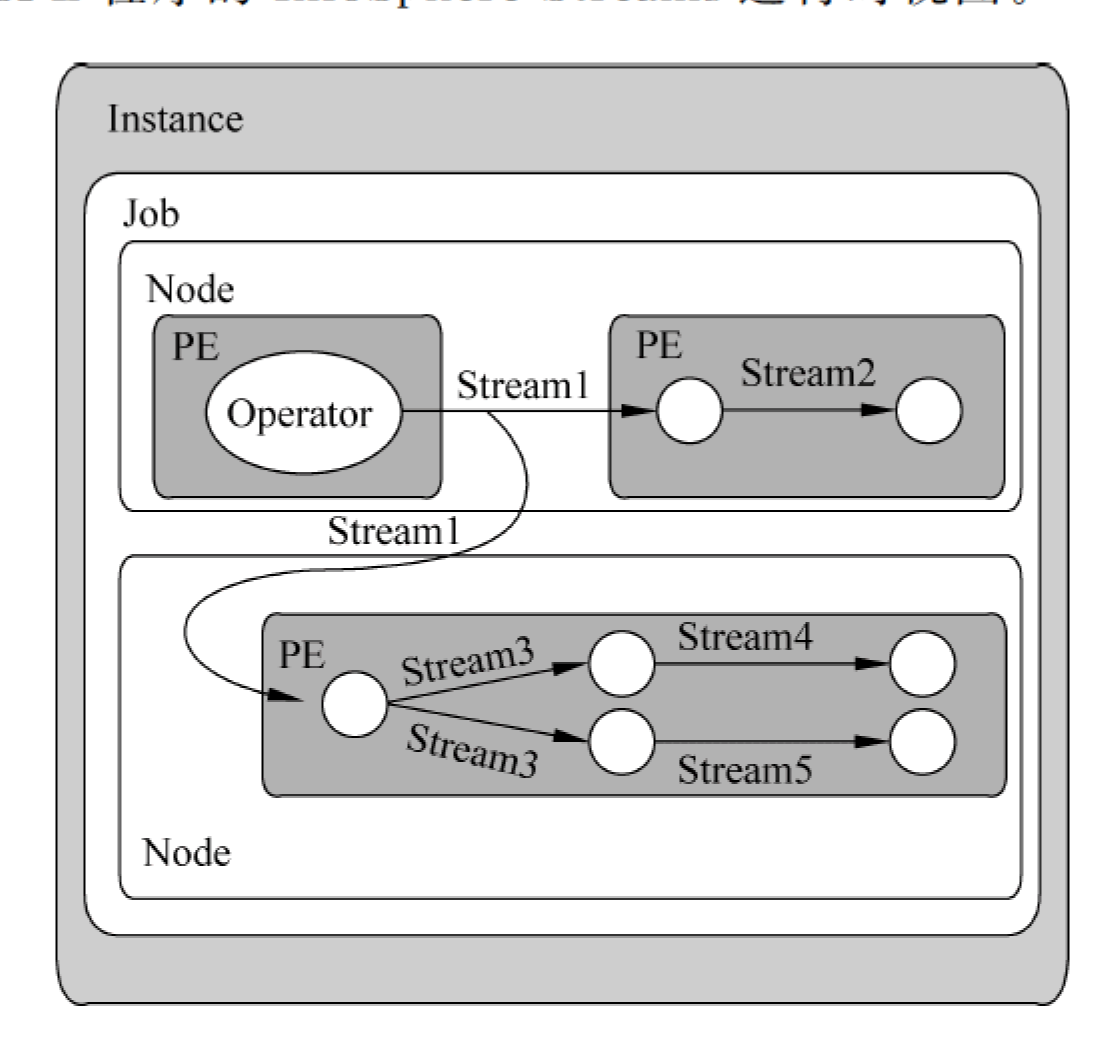


图 5.14 InfoSphere 运行时执行

一个运算符表示一个可重用的流转换器，将一些输入流转换为输出流。在 SPL 程序中，运算符调用可实现预算法的特定用途，使用分配的特定的输入和输出流，以及在本地指定的参数和逻辑。每次运算符调用都会对输入和输出流命名。各种内置的 InfoSphere Streams 运算符提供了许多强大的功能：

- Source：读取流格式的输入数据。
- Sink：将输出流的数据写入外部存储或系统中。
- Functor：过滤、转换和对输入流的数据执行各种功能。
- Sort：对定义的键上的流数据排序。
- Split：将输入流数据拆分为多个输出流。
- Join：合并定义的键上的输入流数据。
- Aggregate：聚合定义的键上的流数据。
- Barrier：组合和匹配流数据。
- Delay：演示一个流数据流。
- Punctor：识别应一起处理的数据分组。

一个流连接到一个运算符的位置称为端口。许多运算符(例如 Functor)有一个输入端

口和一个输出端口，但运算符也可以没有输入端口（比如 Source）和没有输出端口（比如 Sink），或者拥有多个输入或输出端口（比如 Split 和 Join）。清单 1 给出了 Sink 的一个 SPL 示例，它有一个输入端口并将输出元组写入到一个磁盘文件中。

清单 1　Sink 示例

```
() as Sink = FileSink(StreamIn) {
    param
    file : "/tmp/people.dat";
    format : csv;
    flush : 20u;
}
```

在清单 1 中，file 是一个强制性参数，提供了输出文件的路径。flush 参数用于清除给定数量的元组后的输出。format 参数指定了输出文件的格式。

组合运算符是一个运算符集合。它表示对原始（非组合）运算符或组合（嵌套）运算符的一个子图的一种封装。它类似于过程语言中的宏。

一个应用程序由一个没有输入或输出端口的主要组合运算符表示。数据可流入和流出，但不会流到一个图表内的流上，而且流可导出到在同一个实例中运行的其他应用程序和从这些应用程序导入。清单 2 中的代码给出了主要组合运算符的框架。

清单 2　主要组合运算符的结构

```
composite Main {
    graph
    stream ... {
    }
    stream ... {
    }
    ...
}
```

作为一个示例，来看一个简单的流应用程序 WordCount，它统计一个文件中的行数和字数。该程序由以下流图组成：

- 一个 Source 预算法调用，读取一个文件并将各行发送给数据流。
- 一个 Functor 运算符调用，统计行数和每个数据行的字数，将统计数据发送给它的输出流。
- 一个 Counter 运算符调用，聚合文件中所有行的统计数据并打印在末尾。

在介绍 WordCount 的主要组合运算符之前，将定义一些帮助器。为一行的统计数据使用 LineStat 类型。此外，需要构建一个 countWords(rstring line) 函数来统计一行中的字数，需要使用一个 addM(mutable LineStat x, LineStat y) 函数来添加两个 LineStat 值并存储结果。清单 3 定义了这些帮助器。

清单 3　WordCount 帮助器定义

```
type LineStat = tuple< int32 lines, int32 words >;

    int32 countWords(rstring line) {
```

```
        return size(tokenize(line," \t",false));
    }

    void addM(mutable LineStat x,LineStat y) {
        x.lines  += y.lines;
        x.words  += y.words;
    }
```

现在可以定义主要组合运算符了,如清单 4 所示。

清单 4　WordCount 的主要组合运算符

```
composite WordCount {

    graph
    stream<rstring line> Data = FileSource() {
        param file : getSubmissionTimeValue("file");
        format : line;
    }
    stream<LineStat> OneLine = Functor(Data) {

        output OneLine : lines = 1,words = countWords(line);
    }

    () as Counter = Custom(OneLine) {

        logic state : mutable LineStat sum = { lines = 0,words = 0 };
        onTuple OneLine : addM(sum,OneLine);
        onPunct OneLine : if (currentPunct()  == Sys.FinalMarker)

        println(sum);

        }

}
```

5.2.4　开发环境

InfoSphere Streams 提供了一个敏捷开发环境,该环境由 Eclipse IDE、Streams Live Graph 视图和一个流调试器组成。该平台还包含用于加速和简化特定功能或行业的解决方案开发的工具包:

- 标准工具包:包含随产品发布的默认运算符。
  - 关系运算符,比如 Filter、Sort、Functor、Join、Punctor 和 Aggregate。
  - 适配器运算符,比如 FileSource、FileSink、DirectoryScan 和 Export。
  - 实用程序运算符,比如 Custom Split、DeDuplicate、Throttle、Union、Delay、ThreadedSplit、Barrier 和 DynamicFilter。
- 互联网工具包:包括 HTTP、FTP、HTTPS、FTPS 和 RSS 等运算符。
- 数据库工具包:支持 DBMS,包括 DB2®、Netezza、Oracle Database、SQL Server 和

MySQL。

- 其他内置工具包：金融、数据挖掘、大数据和文本工具包。

5.2.5 BigInsights 和 InfoSphere Streams 之间的集成和交互

不断从系统中生成大量宝贵数据的公司正面临为以下两个重要用途而分析数据的问题困扰：及时感知和响应当前事件，根据历史知识进行预测，从而指导响应。这一情形产生了无缝运行移动数据（当前数据）和静止数据（历史数据）分析、处理海量、多样性、高速产生的数据的需求。IBM 的移动数据（InfoSphere Streams）与静止数据（BigInsights）平台的集成解决了 3 个主要应用场景的需求：

- 可伸缩的数据获取：通过 Streams 持续将数据获取到 BigInsights 中。例如，通常需要获取来自社交媒体来源（比如 Twitter 和 Facebook）的非结构化文本数据，以提取各种类型的态度和线索。在这种情况下，如果文本提取在获取数据时执行，那么尽早消除垃圾邮件等无关数据会让效率高很多。这种集成使公司能够避免巨额的非必要存储成本。
- 加速和充实：从 BigInsights 生成历史上下文来加速分析和充实传入的 Streams 数据。BigInsights 可用于分析在较长的时间窗口内从各种连续和静态的数据来源吸收和集成的数据。此分析的结果为各种在线分析提供了上下文，可用于将它们引导至一种已知状态。回到社交媒体应用程序的场景，一条传入的 Twitter 消息仅拥有发布该消息的人的 ID。但是，历史数据可通过属性（比如影响者）充实该信息，为执行下游分析以适当应对此用户所表达的态度提供机会。
- 自适应分析模型：BigInsights 上的分析操作（比如数据挖掘、机器学习或统计建模）所生成的模型。这些可用作分析 Streams 上的传入数据的基础，基于实时观察结果而更新。

IBM 大数据平台的移动数据和静止数据部分可通过 3 种主要的组件类型来集成：

- 通用分析：相同的分析功能可用在 Streams 和 BigInsights 上。
- 通用数据格式：Streams 格式运算符可在 Streams 元组格式和 BigInsights 使用的数据格式之间转换数据。
- 数据交换适配器：Streams Source 和 Sink 适配器可用于与 BigInsights 交换数据。

5.2.6 InfoSphere Streams 环境

Although InfoSphere Streams 可在单台计算机上运行，它被设计成可在集群上运行，以提供几乎无限的可扩展性。一台计算机可用作一台管理主机、一台应用程序主机，或是一台混合模式的主机，这种主机既可以运行管理服务，也可以运行应用服务和代码，如图 5.15 所示。

请注意，管理节点运行的一组服务将会跟踪集群的健康状况和作业的运行情况。可以将一个作业看作是一个程序。作业和程序之间的区别在于它是由操作符构成，可通过调度在任何可用应用主机上运行。作业由多个流程组成，而程序则在单个流程上运行。

InfoSphere Streams 包含的一些工具可帮助用户管理环境和开发 InfoSphere Streams 应用程序。

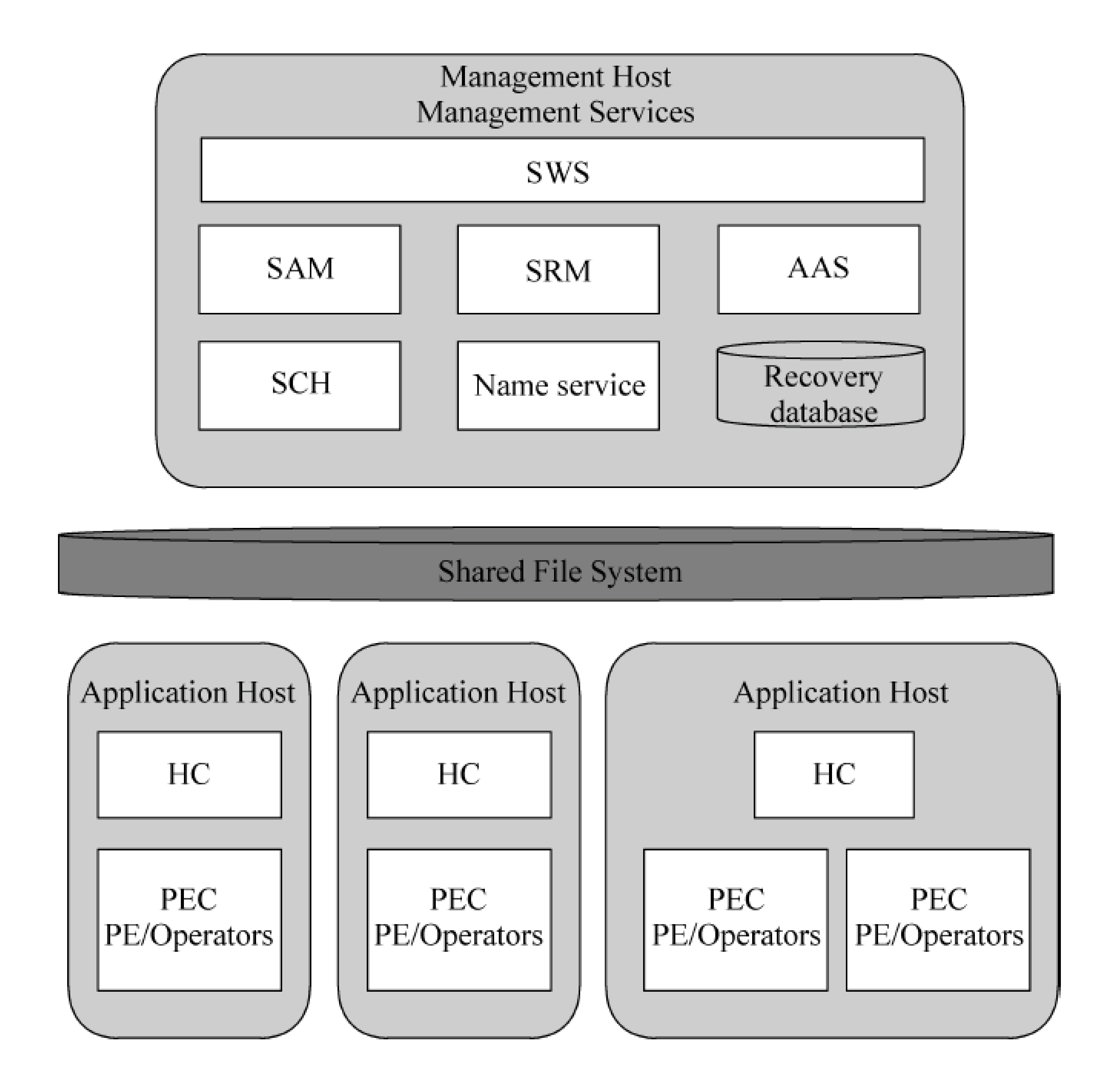


图 5.15 集群上的管理主机和应用主机的混合

- FirstSteps：执行安装后的任务，比如配置 SSH、生成公钥和私钥、配置恢复数据库等。
- Instances Manager：创建、配置、更新和删除实例与集群。实例管理程序还可以启动 InfoSphere Streams 控制台。
- InfoSphere Streams Console：基于 Web 的 GUI，用于监控和管理实例与应用程序。

InfoSphere Streams Studio：基于 Eclipse 的工具，用于开发应用程序。该工具中包含一个图形编辑器和向导，可简化标准开发任务。它还包含运行 InfoSphere Streams 作业及其操作符的可视化。

- Streamtool：命令行工具，用于自动化管理和监控任务。
- InfoSphere Streams Compiler(SC)：通过工具来自动化编译任务。还可直接使用或通过生成文件(makefile)使用。

一旦掌握了图形工具(列表中的前四个符号)，就可以使用它们来帮助大幅提高生产力。学习曲线非常短且值得尝试。

5.2.7 InfoSphere Streams 编程

InfoSphere Streams 编程首先将操作符聚集在一起，以便进行模块化处理并启用并行执行。图形编辑器可以更轻松地将操作符连接在一起，创建一个作业处理图形。

InfoSphere Streams 还使用称为流处理语言(SPL)的程序语言。如果读者熟悉 C 语言、Java 编程语言和 Python，那么应该很快就会熟练使用 SPL。

InfoSphere Streams 与大多数语言不同的其中一个方面是其广泛的可用数据类型集，如图 5.16 所示。

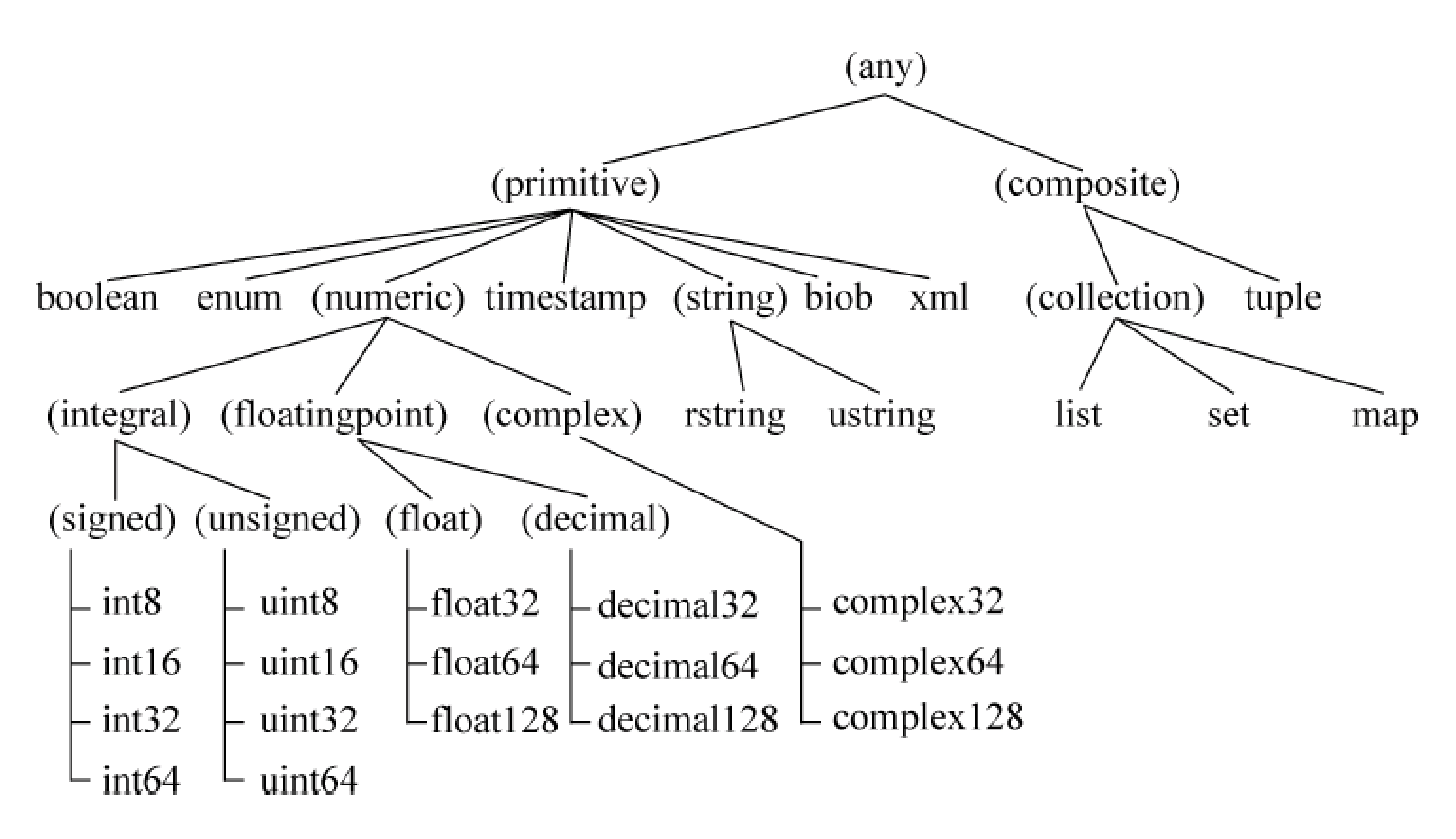


图 5.16　InfoSphere Streams 中可用的数据类型

利用这些类型，几乎可以匹配任何数据类型。甚至可以根据所提供的原始类型创建自己的数据类型。

除了数据类型外，InfoSphere Streams 还附带了重要的分类功能集，包括集合操作（列表、集合与映射）、文件、数学、字符串、时间和实用功能。除了处理不同数据类型的几个同名功能和特定于给定操作符的功能之外，InfoSphere Streams 总共包含二百五十多种功能。用户还可以在 InfoSphere Streams 包含的专用工具集中添加技术功能。

InfoSphere Streams 编程从操作符开始。操作符管理通过各种方式从外部世界获取数据，执行数据转换，并提供最终结果。它支持将框架快速凝聚为一个解决方案，并利用该解决方案来解决业务问题。

5.2.8　操作符和工具集

操作符组成了比程序编程更高级别的抽象层次，利用操作符，就拥有了对处理的逻辑隔离，从而提供了处理分发。操作符来自称为"工具集"的不同数据包。尤其是一个标准工具集始终包含在所有 InfoSphere Streams 项目中。它提供了产品附带的所有功能。尽管各种功能可能会被视为语言的一部分，但还是会被分隔出来，因为它们是在不同的工具集中定义的。

一个工具集就是一个功能数据包，它通常与某个特定的问题领域相关。这些工具集中可以包括操作符、功能和数据类型。下面是 InfoSphere Streams 3.1 中提供的工具集：

- 标准工具集：包含适配器、关系运算、实用程序和 XML 等操作符。
- 大数据：与 Hadoop 分布式文件系统（Hadoop Distributed File System，HDFS）和 Data Explorer 相配合。
- 复杂事件处理：用来定义如何处理复杂事件。
- 数据库：访问关系数据库，比如 DB2、Informix、Netezza、Oracle、SQL Server、Teradata 等。
- 金融服务：包含一组用来处理金融市场处理的操作符和功能。
- 地理空间：提供一组处理地理空间数据的功能。
- InfoSphere DataStage 集成：与 DataStage 相配合。

- Internet：访问 HTTP(S)、FTP(S)和 RSS。
- Messaging：与 WebSphere MQ 和 ActiveMQ 进行通信。
- Mining：提供一种对数据进行评分的方法。
- R Project：与 R Project 统计环境相配合。
- Text：提供文本分析。
- TimeSeries：帮助处理时间序列，即基于时间的数据处理。

标准工具集附带了 36 个操作符以及 256 个以上的功能。其他工具集添加了四十多个操作符和几十个函数。通过所有这些预设的功能，用户可以在创建自定义作业时更有效地满足业务需求。

以上的描述让读者对 InfoSphere Streams 提供的功能有初步的了解。接下来让读者进一步了解 InfoSphere Streams 如何适应完整的数据处理环境。

5.2.9 InfoSphere Streams 集成

InfoSphere Streams 并不是在真空中运行，它是 IBM 大数据平台的关键组件。在企业集成的过程中，InfoSphere Streams 可以从不同来源读取数据并将数据写入不同的目标，比如文件、网络连接(TCP、UDP、HTTP 等)和消息队列(WebSphere MQ 和 Active MQ)。InfoSphere Streams 包含的操作符可与使用金融信息交换(Financial Information eXchange，FIX)协议的金融市场相配合。

除了这些接口之外，InfoSphere Streams 还适用于多种 IBM 产品，包括 InfoSphere BigInsights、Data Explorer、SPSS 和 Cognos。此外，它还可以利用 R Project 统计分析工具。随着 InfoSphere Streams 的发展，还会添加更多的接口。

5.2.10 导航信息中心

信息中心包含的信息适用于系统管理员、InfoSphere Streams 管理员和应用程序编程人员。本节主要介绍应用程序开发人员的信息需求。

图 5.17 显示了信息中心的各个主要部分和子部分。作为一名开发人员，可能想要查看所有部分来了解与 InfoSphere Streams 相关的每件事。

在完成最初的浏览之后，开发人员很可能会花费大量的时间查看参考资料。开发人员需要了解 SPL 语言、可用的功能、可用的操作符以及如何使用它们。

在 Reference|Language reference 中有两个部分：标注查询语言(Annotation Query Language，AQL)和流处理语言(Streams Processing Language，SPL)。如果不打算使用 Text 工具集下的文本分析和 TextExtract 操作符，那么现在可以忽略 AQL 部分。当决定进行文本分析时，用户可能想要使用 InfoSphere BigInsights 2.1 信息中心，其中包含有关各种主题的信息，这些信息比 InfoSphere Streams 信息中心包含的信息还要多。

SPL 语言部分包含了解 SPL 语言、数据类型的使用、功能和如何调用工具集操作符的所有信息。一旦掌握了这种语言，下一个问题就可能是"哪些功能可用来处理我的数据?"这个问题的简要回答是 SPL 不包括任何功能。这可能是真的，但所有应用程序都隐式包含"标准工具集"。本节包含的功能可用于文件处理、数学运算、字符串处理、时间控制和 XML 处理。本小节还包含用于日志记录、跟踪、断言等的实用功能。在查找关于某种功能的信息

时，最简单的方法就是在页面中搜索(使用 Ctrl+F)功能名称或相关关键词。

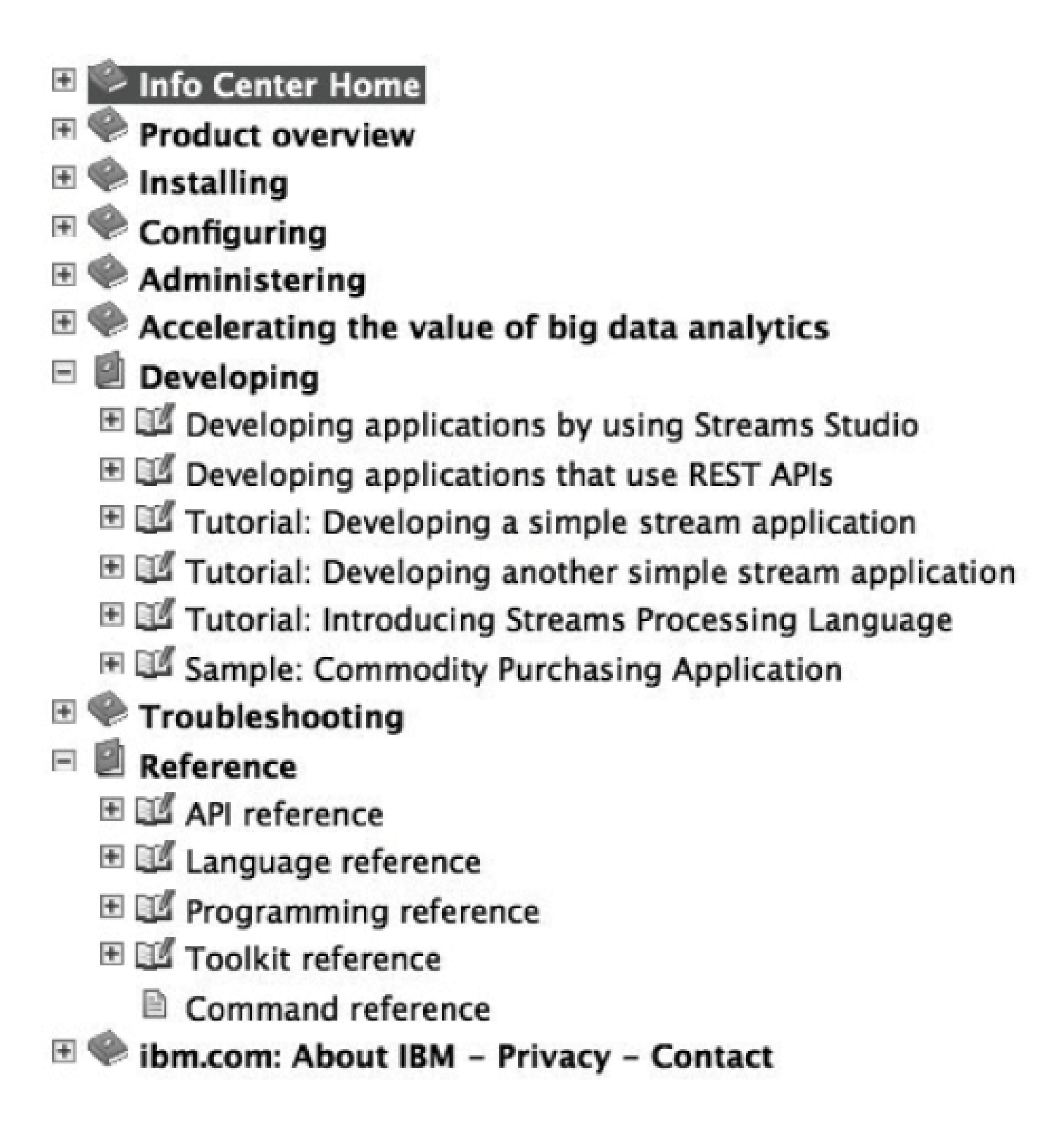


图 5.17　信息中心部分

5.3　IBM InfoSphere BigInsights

5.3.1　IBM InfoSphere BigInsights 简介

BigInsights 是一个分析平台，可帮助公司将复杂的互联网级信息集转换为洞察。它包含一个套装的 Apache Hadoop 发行版(具有高度简化的安装流程)和用于应用程序开发、数据移动和集群管理的关联工具。得益于简单性和可伸缩性，Hadoop(MapReduce 框架的一种开源实现)在行业和学术界获得的巨大的成功。除了 Hadoop 之外，BigInsights 中的其他开源技术(除 Jaql 外的所有技术都属于 Apache Software Foundation 项目)包括：

- Pig：该平台提供了一种高级语言来表达分析大数据集的程序。Pig 配备了一个编译器，可将 Pig 程序转换为 Hadoop 框架执行的 MapReduce 作业序列。
- Hive：一个构建于 Hadoop 环境之上的数据仓库解决方案。它为 Hadoop 的非结构化世界带来了人们熟悉的关系数据库概念，比如表、列和分区，以及 SQL 的一个子集(HiveQL)。Hive 查询被编译为使用 Hadoop 执行的 MapReduce 作业。
- Jaql：IBM 专为 JSON(JavaScript Object Notation，JavaScript 对象表示法)开发的一种查询语言，提供了一种类似 SQL 的接口。Jaql 适度地处理嵌套，高度面向函数，而且非常灵活。它适用于松散的结构化数据，是 HBase 列存储和文本分析的接口。
- HBase：一个面向列的 NoSQL 数据存储环境，旨在支持 Hadoop 中大型、稀疏填充的表。
- Flume：一种分布式、可靠且可用的服务，用于高效地移动生成的大量数据。Flume

非常适合从多个系统中收集生成的日志，为它们插入 HDFS(Hadoop Distributed File System，Hadoop 分布式文件系统)。

- Lucene：一个搜索引擎库，提供了高性能的、全功能的文本搜索。
- Avro：一种数据序列化技术，使用 JSON 来定义数据类型和协议，以一种紧凑的二进制格式对数据执行序列化。
- ZooKeeper：一种维护配置信息和命名，提供分布式同步和分组服务的集中化服务。
- Oozie：一个工作流调度程序系统，用于管理和编排 Apache Hadoop 作业的执行过程。

此外，BigInsights 发行版还包含以下 IBM 独有的技术：

- BigSheets：一种基于浏览器、类似电子表格的查询和探索接口，使业务用户能够轻松地收集和分析数据，利用 Hadoop 的强大功能。它提供了内置的阅读器，可处理多种常见格式的数据，包括 JSON、逗号分隔值(CSV)和制表符分隔值(TSV)。
- Text analytics：常见业务实体的文本注释符的一个预先构建的库。它提供了丰富的语言和工具来构建自定义位置注释符。
- Adaptive MapReduce：一个 IBM Research 解决方案，通过更改 MapReduce 任务的处理方式来加速小型 MapReduce 作业的执行。

5.3.2 IBM InfoSphere BigInsights 3.0 介绍

如图 5.18 所示，IBM 在 Hadoop 开源框架的基础上进行了大量的开发和扩展，陆续将丰富的高级文本分析(Advanced Text Analytics Toolkit，研发代码 SystemT)、机器学习(Machine Learning Toolkit，研发代码 SystemML)、GPFS File Place Optimizer(GPFS-FPO)、IBM LZO 压缩、针对 Jaql 的 R 模块扩展、改进的工作负载调度(Intelligent Scheduler)、自适应 MapReduce(Adaptive MapReduce)、基于浏览器的可视化工具(BigSheets)、大规模索引、搜索解决方案构建框架(BigIndex)、统一的 SQL 接口(BigSQL)、大规模并行处理 SQL 引擎(MPP SQL Engine，IBM Big SQL)等纳入到 InfoSphere BigInsights 中，并增强了高可用性、可扩展性、安全性、易用性、监控和告警等，通过支持 LDAP 身份验证增强安全性(另外还能够提供可插拔身份验证支持，支持 Kerberos 等其他协议)，构建了一个完整的企业级大数据平台。该平台为开发人员提供了全面的开发和运行时环境来构建高级分析应用程序，为企业用户提供了完善的分析工具来分析大数据，从而使与大数据分析相关的时间价值曲线变平。

InfoSphere BigInsights 是企业级的海量大数据存储和分析平台，支持对结构化、半结构化和非结构化海量数据的存储和快速分析，提供多节点的分布式计算，可以随时增加节点提升数据处理能力，通过集成的安装程序简化安装流程，通过与 IBM 数据分析软件的深度集成为企业用户提供强大的分析能力，并提供开放性接口和集成能力。BigInsights 作为基于企业级的海量大数据存储和分析平台，并不是对数据仓库的替代，而是对传统数据仓库的一种补充和延伸，和数据仓库一起构成了更广泛的大数据平台。BigInsights 以 Apache Hadoop 及其相关开源项目作为核心组件，IBM 将继续保持开源项目的完整性，防止与核心分离或偏离，BigInsights 3.0 中 Hadoop 开源组件所对应的版本如表 5.1 所示。

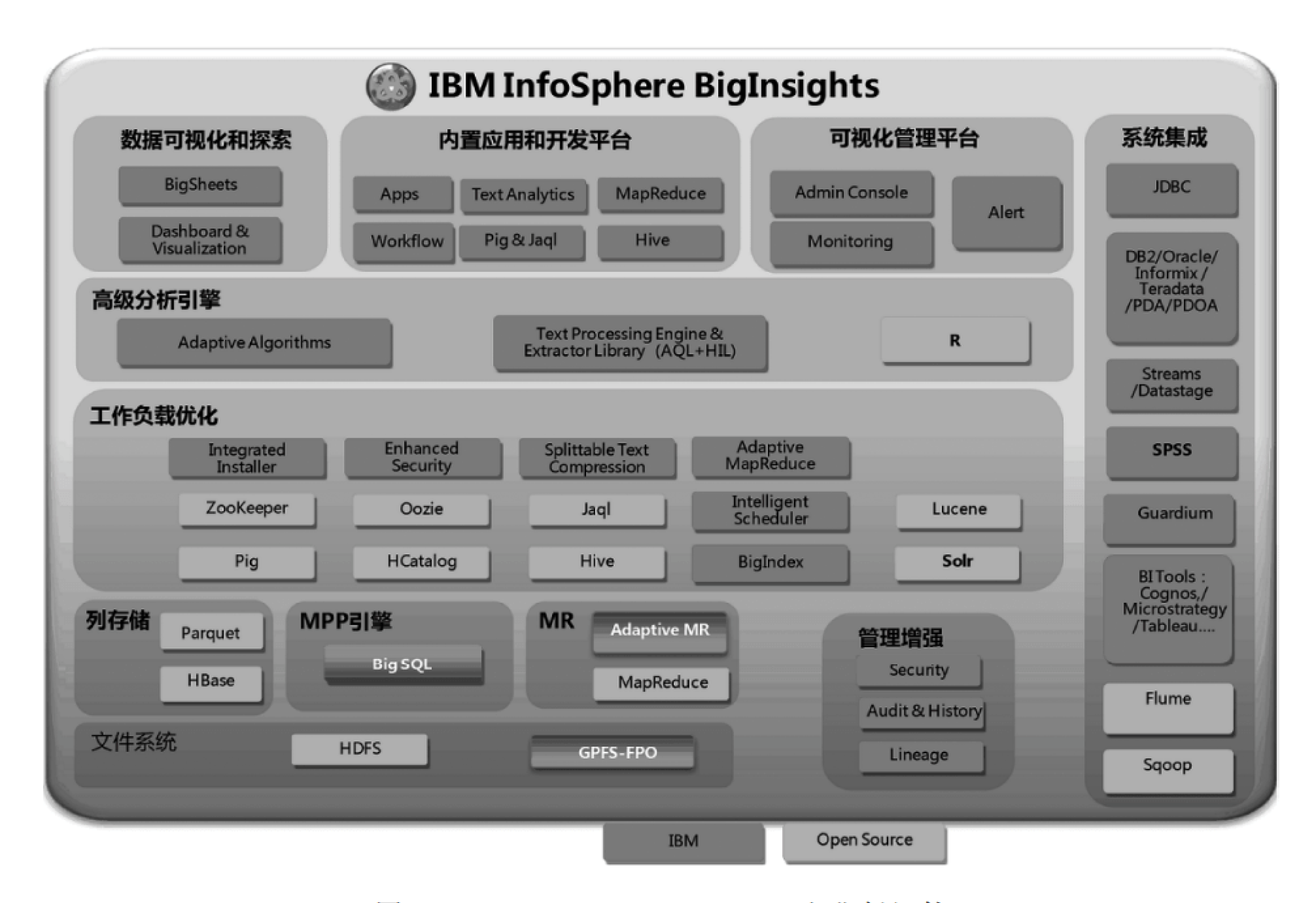


图 5.18 InfoSphere BigInsights 企业版组件

表 5.1 BigInsights 开源组件对应版本

Component	BigInsights 2.1	BigInsights 2.1.2	BigInsights 3.0
Hadoop	1.1.1	2.2.0	2.2.0
HBase	0.94.3	0.96.0	0.96.0
Hive	0.9.0	0.12.0	0.12.0
Pig	0.10.0	0.12.0	0.12.0
Zookeeper	3.4.5	3.4.5	3.4.5
Oozie	3.2.0	3.3.2	3.3.2
Hcatalog	0.4.0	Now part of Hive 0.12.0	Now part of Hive 0.12.0
Sqoop	1.4.2	1.4.3	1.4.3
Flume	1.3.0	1.3.1	1.3.1
JAQL	2.0.0	2.0.0	2.0.0
Lucene	3.3.0	3.3.0	4.7.0
Solr	NA	NA	4.7.2.1
Avro	1.7.2	1.7.2	1.7.4

5.3.3 IBM Big SQL 3.0

IBM Big SQL 3.0 是一个大规模并行处理 SQL 引擎(MPP SQL Engine),可以直接部署在物理的 HDFS 集群上。通过使用一个低延时并行执行基础架构,并将处理操作放在数据所在的节点,Big SQL 3.0 实现了 native 方式的 Hadoop 数据访问,包括读和写操作。Big SQL 3.0 数据库基础架构提供一个所有数据的逻辑视图(通过 Hive 元数据管理),查询编译视图,以及为最优的 SQL 处理提供优化和运行时环境。针对复杂嵌套的决策支持查询,

Big SQL 3.0 专门做了优化处理。Big SQL 3.0 接口完全支持 SQL 2011 标准、支持存储过程、用户自定义函数、广泛的数据类型、JDBC/ODBC 接口等。

Big SQL 所有的数据都保持原有的格式存储在 Hadoop 集群中，Big SQL 对数据的格式没有特殊的要求，Hadoop 表的定义是由 Hive Metastore 定义和共享的，使用 Hadoop 的标准目录。Big SQL 共享 Hive 接口进行物理读取，写入和理解 Hadoop 中存储的数据。简单来说，Big SQL 中没有任何的数据存储，数据是存放在 Hadoop 集群中的，在 Big SQL 中定义的表实际上是一个在 Hive 中定义的表。通过 Apache HCatalog，Hive 中定义的表可以作为很多工具的有效数据源。最终，任何 Hadoop 应用程序都可以访问这些大规模共享的分布式文件系统中的简单文件。

Big SQL 既可以运行在 POWER Linux (Red Hat)上，也可以运行在 x64 Linux 上，如 Red Hat 和 SUSE Linux。

Big SQL 3.0 汇集了成熟高效的 IBM SQL 编译器和运行时引擎，通过与 Hive 数据库目录和 I/O 层一起工作，允许用户执行 SQL 2011 标准的查询，并保证企业级的性能。通过引入 SQL PL 兼容性，Big SQL 3.0 相比 BigInsights 2.1 中 Big SQL 包含的功能，已经进行了大大的扩展，具体包括存储过程、SQL-bodied 函数和丰富的 scalar 库、表和 OLAP 函数等。

5.3.4 企业集成

如图 5.19 所示，通过 Big SQL 3.0 的联邦功能，用户可以对很多数据源如 DB2、Oracle、Teradata、IBM PureData System for Analytics(PDA)、IBM PureData System for Operational Analytics(PDOA)等进行联邦访问。联邦功能允许用户在同一个 SQL 语句内给各个关系型数据源发送分布式的请求。通过 Big SQL 3.0 提供的丰富 SQL 功能，很多 BI 工具如 IBM Cognos BI、Microstrategy 和 Tableau 等可以方便地访问 Hadoop 集群中的数据并执行各种处理。

BigInsights 3.0 可以集成多种产品，支持与大量数据源进行数据交换(比如 DB2、Teradata、PureData for Analytics 等)，更多的数据源可以通过 JDBC、InfoSphere Steams、InfoSphere DataStage 和 Sqoop 等进行数据交换，另外 BigInsights 支持与 IBM InfoSphere Data Explorer (Vivisimo)、InfoSphere Guardium、Cognos Business Intelligence、SPS Analytic Server、WAS 8.5 Liberty Profile 和 Rational & Data Studio 等集成。

BigInsights 可以和 InfoSphere Data Explorer(联邦发现和导航工具)进行索引和界面集成，通过索引可以实现联邦访问，通过界面集成用户可以方便地同时使用两个产品。另外，Cognos BI(特别是 Social Media Analysis)和 InfoSphere Streams 也和 BigInsights 进行了界面集成。通过连接器，BigInsights 可以和 PureData System for Analytics 进行双向数据交换，BigInsights 3.0 中包含 PureData System for Analytics UDFs，用户编写的 PureData System for Analytics 应用程序可以使用这些 UDF 访问大数据和运行大数据操作。同样通过高速连接器，BigInsights 可以并行访问 IBM DB2、Teradata、InfoSphere Warehouse、PureData System for Transactions 和 PureData System for Operational Analytics。通过 JDBC 连通性(Jaql 的 JDBC 接口)，BigInsights 可以访问更多的数据源。

DataStage 除了改进与 HDFS 相关的集成，还新增了与通用 MapReduce 作业相关的集

成，并和 BigInsights 进行了紧密结合。BigInsights DataStage 连接器已经与 HDFS 和 GPFS-FPO 实现了全面的集成，可以充分利用集群架构的优势，以便将所有批量数据并行写入同一文件。通过与 DataStage 集成，BigInsights 可以成功和绝大多数软件产品实现快速数据交换。

通过 Hive 或 Big SQL 方便用户在 Cognos BI 中使用存储在 BigInsights 中的数据。通过 R 包装器，BigInsights 3.0 可以帮助人们以集群方式运行 R 应用程序(BigInsights 则通过 Jaql 对 R Project 进行了扩展，使其可以分布式并行运行。用户可以基于 Jaql MapReduce 运行 R 查询)。在 BigInsights 3.0 中还包含与 WebSphere 8.5 Liberty Profile 集成，提供高性能、安全的 REST 访问。另外，BigInsights 3.0 还支持与 RAD 和 Rational Team Concert & Data Studio 协作和开发整合。

通过 IBM SPSS Analytic Server(IAS)，用户可以方便地在 SPSS Modeler 客户端或者 SPSS Analyitc Catalyst 中基于 BigInsights 中的数据进行数据挖掘建模。

图 5.19 企业集成

如图 5.20 所示，IBM SPSS Analytic Server 是处于客户应用程序和 Hadoop 集群中间的数据分析引擎。用户通过使用 SPSS Modeler 客户端或者 SPSS Analyitc Catalyst，将各种分析请求发送给 SPSS Analytic Server，SPSS Analytic Server 将协调作业将其运行在 Hadoop 集群并将结果返回给客户端应用程序。

针对 BigInsights 的审计和监控还可以使用 IBM InfoSphere Guardium。通过使用 IBM InfoSphere Guardium，用户可以获取有针对性的、可操作的信息，极大地简化了用户审计过程。通过定义安全策略，用户可以指定需要保存什么数据以及如何应对策略违规。数据事

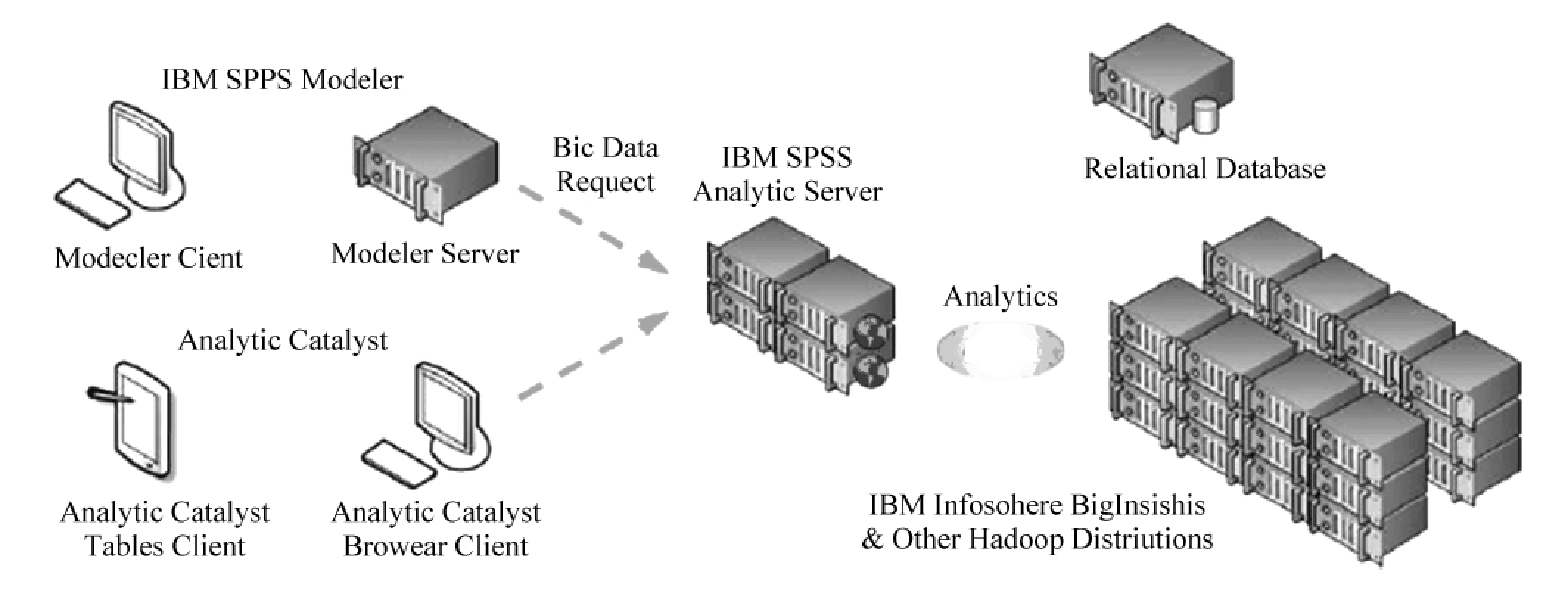


图 5.20 IAS 体系结构

件直接写入 InfoSphere Guardium 收集器，特权用户甚至都没有机会访问并隐藏他们的踪迹。开箱即用的报告可以让用户立即开始快速运行 BigInsights 监控，而且这些报告可以很容易通过定制来符合用户的审计需求。IBM InfoSphere Guardium 主要使用探测器（称为 S-TAP，用于软件）对 BigInsights 进行监控，无须依赖 BigInsights 的审计日志即可监控所有相关操作，无须对系统软件或应用程序进行任何更改。IBM InfoSphere Guardium 对 BigInsights 监控的事件包括：

- 会话和用户信息。HDFS 操作：命令（cat、tail、chmod、chown、expunge，等等）。
- MapReduce 作业：作业、操作、权限。
- 异常：比如授权故障。
- Hive/HBase 查询：改变、计数、创建、删除、获取、放置、列出，等等。

另外 BigInsights 中还包括一个集成功能，即 Guardium Proxy，可读取日志消息并发送到 InfoSphere Guardium 用于分析和报告。有了这个代理，BigInsights 就可以将消息从 Hadoop 日志发送到 InfoSphere Guardium 收集器。该代理的优势包括以下几点：

- 易于启动和运行。不需要安装 S-TAP 或配置端口。只需要在 NameNode 启用代理就可以使用了。
- 由于代理使用 Apache 日志数据作为消息发送到 InfoSphere Guardium，所以需要从消息（例如状态和心跳信息）中过滤的噪音就很少。
- Guardium 对支持新版的 BigInsights 利用消息协议变更不存在延迟。

限制：由于 Hadoop 没有将异常记录到其日志中，所以没有办法将异常发送到 InfoSphere Guardium。如果需要异常报告，那就需要实现一个 S-TAP。除此之外，不支持监控 HBase 或 Hive 查询，但可以从 Hive 和 HBase 看到基础的 MapReduce 或 HDFS 消息。

5.3.5 GPFS File Place Optimizer

GPFS File Place Optimizer（GPFS-FPO）之前也被称为 GPFS 无共享集群文件系统（General Parallel File System-Shared Nothing Cluster，GPFS-SNC），是 IBM 2009 年在 GPFS 的基础上扩展而来的，使其能够处理带有 GPFS-FPO 的 Hadoop（GPFS 最初只能作

为存储区域网络(SAN)文件系统使用)。GPFS-FPO 遵循 POSIX,没有主控的 NadmeNode 节点,而是将元数据分散到集群节点中,避免了单点故障。GPFS-FPO 比 GPFS 增加了局部性认知功能、元数据块、写入关联和可配置的复制和可配置恢复策略等。由于 GPFS-FPO 是 GPFS 的扩展,本身还是 GPFS,所以在 GPFS-FPO 中可以实现同样的稳定性、灵活性和性能,另外 GPFS-FPO 通过分层存储管理(HSM)以不同的检索速度管理和使用磁盘驱动器,管理不同热度的数据,保证数据位于最近硬件上。

局部性认知功能:负责提供集群中文件位置给 JobTracker 以便 JobTracker 根据这些位置信息选择需要运行的本地任务副本以提高性能。

元数据块:典型的 GPFS 数据块大小一般是 256KB 大小,而 Hadoop 中的数据块通常比这个大得多,比如 BigInsights 中建议数据块大小为 128MB,开源 Hadoop 中数据块大小默认是 64MB,为了满足大数据块的使用需求,GPFS-FPO 通过将多个 GPFS 数据块叠加在一起组成更大的数据块,也就是元数据块,方便运行 map 任务;对 Hadoop 以外的文件还使用较小的数据块,从而确保了各种应用程序可以在一个集群中运行,保证了整个集群的性能,HDFS 则不具备这样的优点(非 Hadoop 的文件无法直接在 HDFS 集群上运行,需要在本地文件系统而非 HDFS 集群上运行这些文件),比如 BigIndex 或 Lucene 全文索引在 GPFS-FPO 中可以运行,而 HDFS 则不行。

写入关联和可配置的复制:允许为文件定义位置策略,默认情况下复制策略为第一个副本为计算机本地副本,第二个副本是机架本地副本,第三个副本则以条带形式分布在集群中的其他机架之间(HDFS 不一样,一般情况 HDFS 其他两个副本会在另外(远程)机架上的不同节点上,且 HDFS 不支持条带化)。可以指定一组特定的文件始终存储在一起,以便应用程序从同一个位置读取数据(HDFS 无法做到)。

HDFS 会使用 NameNode 节点统一存储和维护元数据,GPFS-FPO 则不需要 NameNode 或任何类似的硬件充当元数据中央存储区,元数据在集群中多个节点间共享(分布式元数据),通过大量数据块随机读取提供了性能,避免了 HDFS 中访问元数据集中存储的 NameNode 造成性能瓶颈。另外,通过客户端缓存,GPFS-FPO 提升了随机工作负载的工作性能。GPFS-FPO 是一种内核级文件系统(这点和 HDFS 不同),全面支持 POSIX,应用程序可以方便地查看 GPFS-FPO 中存储的文件,并对文件进行各种操作。在 GPFS-FPO 中,MapReduce 应用程序或其他应用无须追加内容即可更新现有的集群文件,还能保证多个应用程序并行写入 Hadoop 集群中的同一文件(HDFS 无法实现上述功能)。在 GPFS-FPO 集群中有多种管理角色承担不同的职责,比如仲裁节点(Q)、集群管理器(CM)、主集群配置服务器(P)、辅助配置服务器(S)、文件系统管理器(FSM)节点和元节点(MN)等。

5.3.6 IBM Adaptive MR

IBM Adaptive MR 是一个优化的 MapReduce 运行时环境,通过使用 C++/C 重写了 Job Tracker/Task Tracker,采用松耦合的 resource manager 和 Job trackers,采用 SOA 架构,实现了数据存储和共享服务,减少了多个 tasks 之间的 JVM 交互,使用原生的 TCP 协议和基于 marshalling 的二进制传输格式,代替 http 和 text/XML,task tracker 使用 Push 协议,避免 task tracker pull task 时由于轮训执行 pull 的间隔而带来的延迟。图 5.21 为 IBM Adaptive MR 运行时环境。

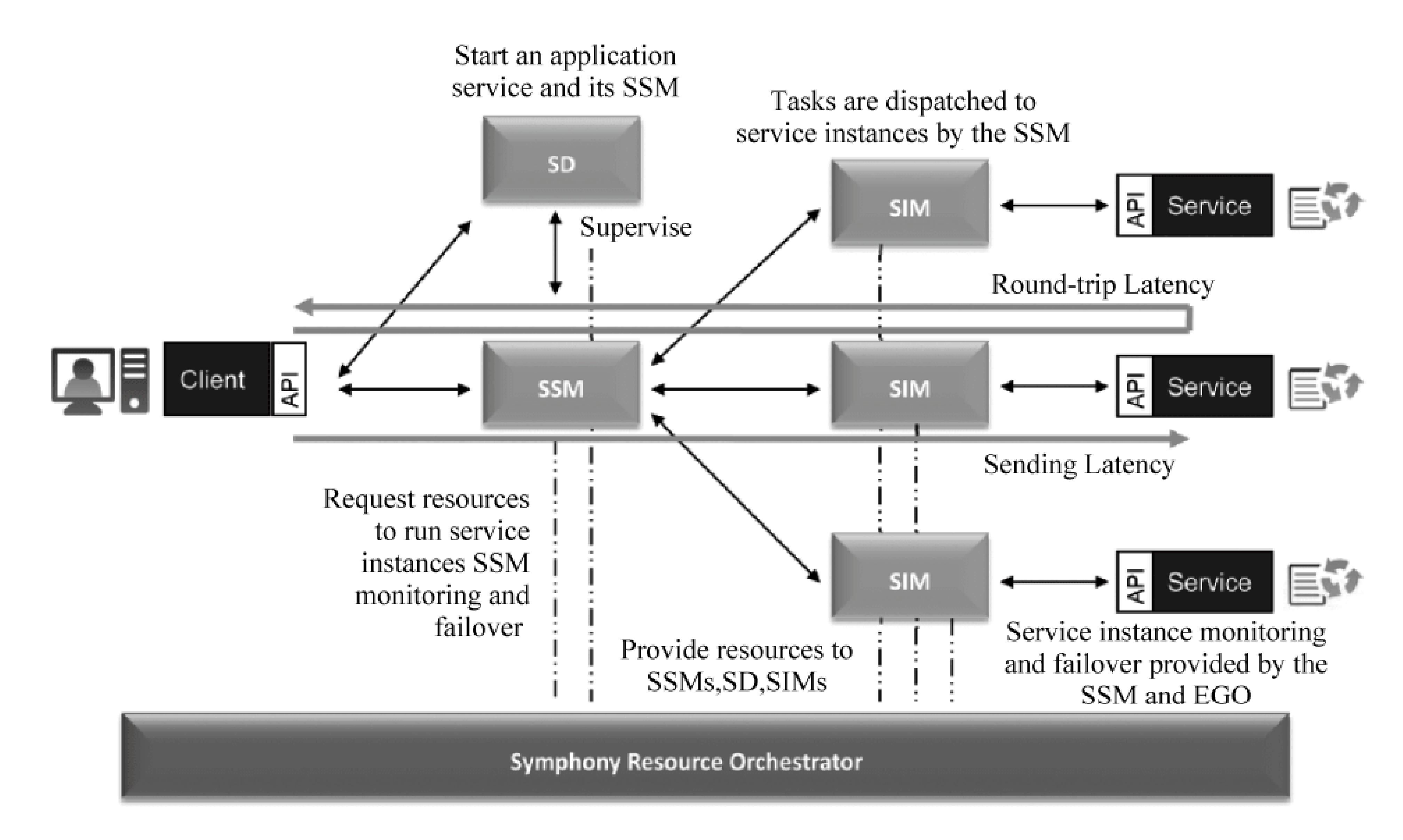


图 5.21 IBM Adaptive MR 运行时环境

使用 Apache MapReduce 执行作业时，Shuffle 需要将 Map 输出的数据写入到硬盘，再将数据复制到 Reduce 节点，而经过 BigInsights 优化的 Shuffle 采用将 Map 输出的数据尽量保持在内存，不能保持时再写入数据到硬盘，Reduce 节点则尝试从内存中获取数据。同样在输入数据时，经过 BigInsights 优化的 Shuffle 也尽量保持在内存，不能保持时再写入到 Reduce 节点的硬盘。数据都保持在内存中，明显提升了 Shuffling 效率。

5.3.7 IBM BigSheets

BigSheets 是 BigInsights 提供的数据发现和可视化分析工具之一。Hadoop 本身使大数据分析成为可能，编程人员通过使用 mapReduce 编程探索数据，但就像传统数据仓库构建方式一样，开发人员如果熟悉数据仓库也可以基于编程的方式实现运营分析，可对大多数业务用户以及管理用户来说非常不方便，BigSheets 就是 BigInsights 提供的基于浏览器的电子表格风格的大数据分析工具，用户无须编程就可使用 BigSheets 对海量数据实施分析，无论采用哪种数据结构。如图 5.22 所示，当数据收集完成后，用户可以在电子表格界面查看数据样本和操控数据，比如可以合并不同集合的列，运行公式或过滤数据等。

如图 5.23 所示，在 BigSheets 中，将使用 master workbooks（主工作簿）、workbooks（工作簿）和 sheets（表）采集数据。master workbook 主要负责从一个输出结果文件中获取数据，进行数据采集的初始化工作，master workbooks 中的数据是只读的，用户可以基于最初的原始数据浏览数据集。输出结果文件可以通过上传一个文件或通过使用应用程序收集数据方式来创建。在 master workbook 中通过一个映射（map）或图表（chart）可以进行数据的简单可视化，如果想进一步探索数据，需要在 master workbook 的基础上创建新的 workbooks。workbooks 包含的数据来自一个或多个 master workbooks 或 child workbooks，用户可以定制数据的格式、内容和结构。可以创建 workbooks 来保存一组特定的数据结果集，重定义数据和探索数据。可以基于一个 master workbook 或其他的

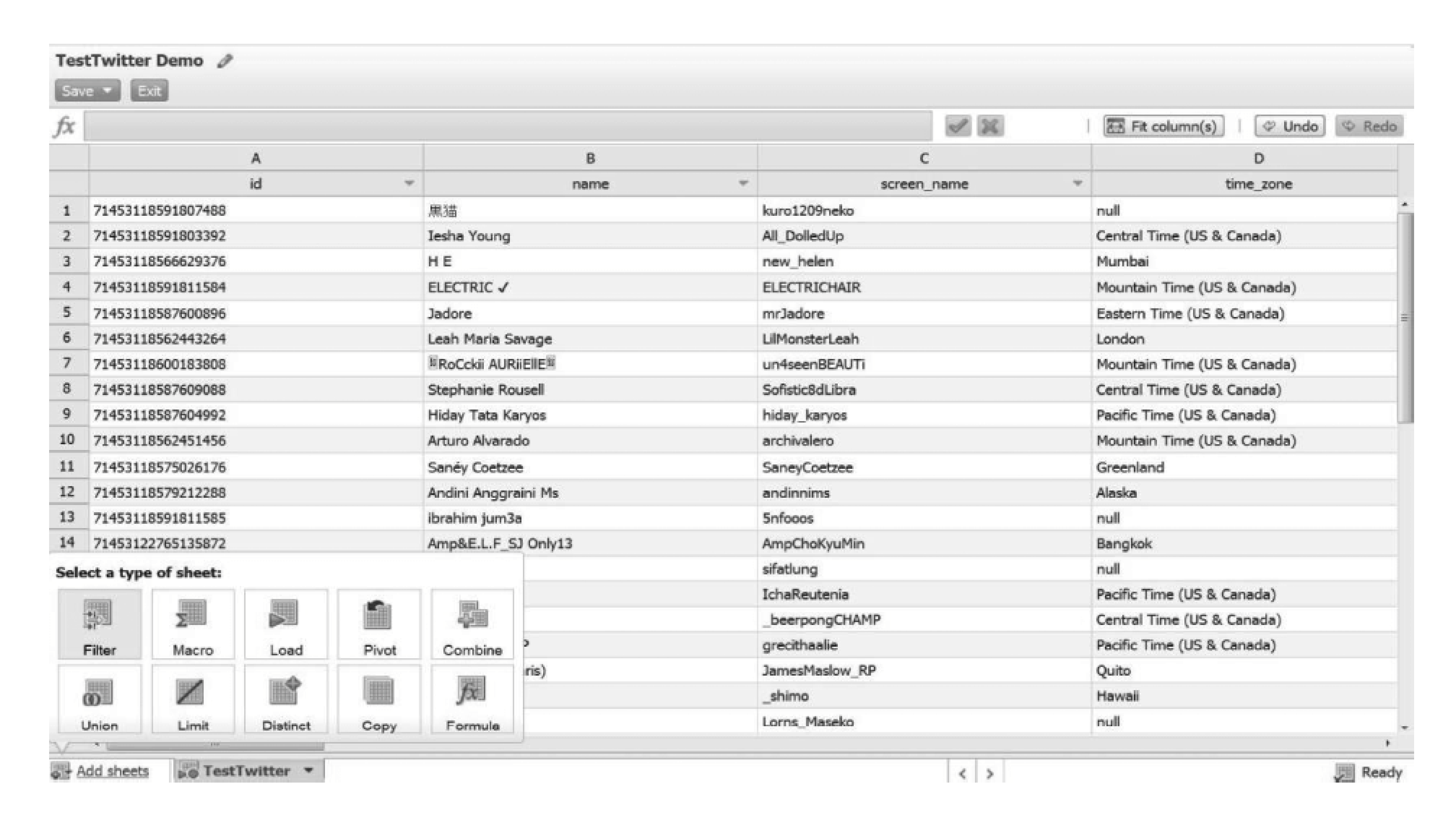


图 5.22　电子表格风格的 BigSheets

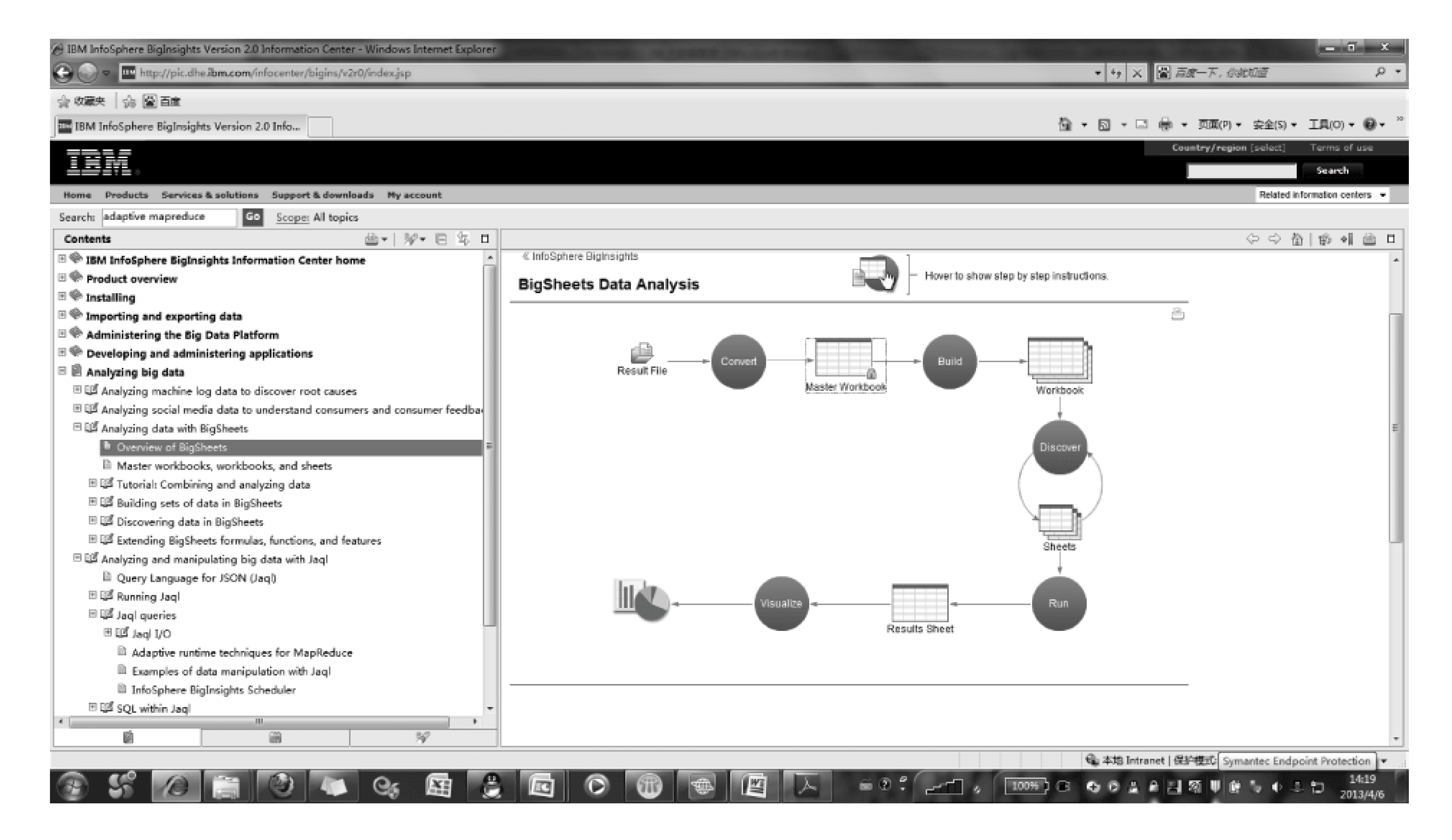


图 5.23　BigSheets 概述

workbook 创建 workbooks，如果 workbook 是基于一个 master workbook 创建的，那么该 workbook 被称为该 master workbook 的 child。如果 workbook 是基于另外一个 workbook，新的 workbook 则是已经存在的 workbook(parent workbook)的 child。在这些 workbooks 和它们所有的后代之间就建立了工作簿相关关系。workbooks 可以拥有一个或多个 sheets，sheets 是应用不同的功能分析和查看的数据子集的数据表示，sheet 中的每一行表示数据的一条记录，每一列表示该记录的一个属性。可以在 workbooks 中添加 sheets 逐步编辑和浏览数据。默认情况下，在 workbooks 中的最后生成的 sheet 被称为 Result sheet，当保存和运行 workbook 时，Result sheet 中的数据是 workbook 的输出或结果。用

户可以基于 Results sheet 创建 maps 或 charts 来可视化结果数据。

如图 5.24 所示，在 BigSheets 2.0 中，集中式的仪表盘允许业务分析人员利用新的图形引擎使用 BigSheets(类似电子表格的可视化工具)，基于数据获得洞察，查看分析应用程序结果和监控指标等。



图 5.24　集中式的仪表盘

5.3.8　高级文本分析

文本分析技术对大数据分析和探索非常重要，可以帮助用户进行日志分析、电子邮件分析(比如用于欺诈分析)、社交媒体分析(比如评估客户情绪等)以及其他各种文本相关分析。BigInsights 纳入了高级文本分析(Advanced Text Analytics Toolkit，研发代码 SystemT)功能，用于读取非结构化文本和提炼洞察。Advanced Text Analytics Toolkit 的核心是 Annotator Query Language(AQL)，这是一种全声明性文本分析语言(没有“黑盒”)，所有模块均可自定义，也就是说，所有数据都采用相同语义进行编码，并遵循相同的优化规则。AQL 提供类似 SQL 语言用于构建提取程序。通过文本分析纳入到应用程序，可以读取非结构化的文本，并从信息中获取洞察。如图 5.25 所示，详细的文本分析生命周期包括开发文本分析 extractors，运行 extractors 和可视化分析结果等。

5.3.9　Solr

Solr 是一款受到广泛欢迎的开源高性能企业搜索平台，是基于 Apache Lucene 项目开发的，其主要的功能包括强大的全文检索、命中标示(hit highlighting)、层面搜索(faceted search，即统计)、近实时的索引、动态聚类、数据库整合、丰富的文档处理能力(比如 Word、PDF 等)和地理位置搜索(geospatial search)等。Solr 具有非常好的可靠性、可扩展性和容错性，提供分布式索引、复制和负载均衡查询、自动故障转移和恢复、集中配置等等。Solr 帮助世界上许多大型互联网网站加强其搜索和导航功能。

Solr 是基于 Java 编写并作为一个独立的全文搜索服务器运行在一个 servlet 容器中(如 Jetty)。Solr 使用 Lucene 的 Java 搜索库进行全文索引和全文检索，并拥有 XML/

HTTP 和 JSON/Python/Ruby 等 APIs，方便用户使用各种编程语言访问 Solr。Solr 具有以下特点：

- 高级全文检索功能；
- 为大容量的网络流量提供优化；
- 基于标准开发接口：XML、JSON 和 HTTP；
- 全面的 HTML 管理界面；
- 通过 JMX 为监控提供服务器统计信息；
- 线性扩展、自动索引复制、自动故障转移和恢复；
- 近实时的索引；
- 灵活和自适应的 XML 配置；
- 可扩展的插件体系结构。

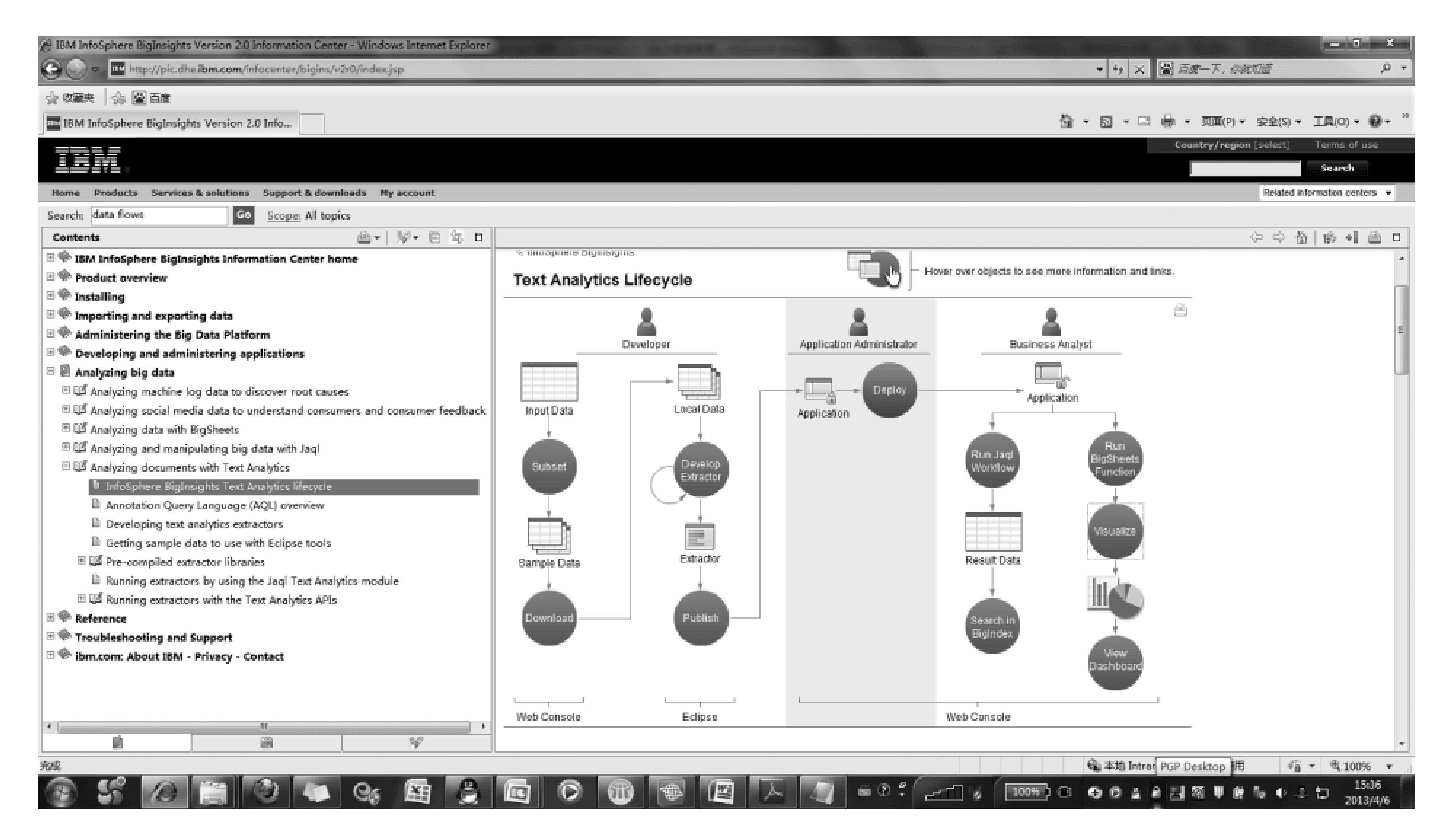


图 5.25　文本分析生命周期

5.3.10　改进工作负载调度

改进工作负载调度(Intelligent Scheduler)之前被称为 Flexible Scheduler 或 FLEX 调度器，是在 Fair Scheduler 基础上扩展而来，通过不断调整执行作业的插槽最低数量来进行操控。Intelligent Scheduler 通过使用各种指标来完成工作负载优化，用户可以根据整个集群的情况选择这些指标。开源 Hadoop 中自带了先进先出(FIFO)基础调度器以及支持替代方案的可插拔架构：Fair Scheduler 和 Capacity Scheduler，这两种工具都是为小型作业提供最低水平的资源以免资源匮乏，无法提供更多更全面的控制以实现整个集群的最佳性能，也不提供足够的灵活性供管理员自定义工作负载。

BigInsights 包含 Fair Scheduler(不包含 Capacity Scheduler)并将其扩展成 Intelligent Scheduler(也是 BigInsights 中默认的调度程序)，Intelligent Scheduler 为 MapReduce 作业提供一个自适应(灵活的)的工作流程分配方案，基于用户选择的策略进行优化处理，保证集

群内所有工作随着时间推移都能得到一个公平的集群资源份额。

启动 Intelligent Scheduler 需要在 mapred-site.xml 文件中添加下列属性：

```
<property>
 <name>jobtracker.taskScheduler</name>
 <value>com.ibm.biginsights.scheduler.WorkflowScheduler</value>
</property>
```

Intelligent Scheduler 相关的指标(metric)或 algorithm(算法)、类优先级配置是对每个集群有效，更改 Intelligent Scheduler 或 JobTracker 指标(metric)需要重启。Intelligent Scheduler 默认调度指标是平均响应时间指标(average response time metrics)，Scheduler 将给小型工作分配最大资源从而保证这些作业尽快完成。

Intelligent Scheduler 还支持 Fair metric 和 Max Stretch metric，当使用 Fair metric 时，scheduler 将模拟 Hadoop Fair scheduler 的行为，Max Stretch metric 则按作业所需的资源量比例为各作业分配资源，也就是说大型作业具有较高的优先级。调度指标和相关的调度算法可以在 mapred-site.xml 文件中进行指定：

```
<property>
 <name>mapred.workflowscheduler.algorithm</name>
 <value>AVERAGE_RESPONSE_TIME</value>
<!-- Possible values are :
 <value>AVERAGE_RESPONSE_TIME</value>
 <value>MAXIMUM_STRETCH</value>
 <value>FAIR</value>
 The default is AVERAGE_RESPONSE_TIME
 -->
</property>
```

Intelligent Scheduler 还可以根据作业优先级给作业分配相应的资源。通过配置 Hadoop 中 JobConf 的 flex.priority 属性，可以根据每个作业或每个 Jaql 查询指定优先级。该属性值是一个数字，用来标识作业所属的优先类(priority class)。默认情况下有 3 个优先类，也就是说 flex.priority 可以被设置为 0，1 或 2，默认值为 2。比如，在一个 Jaql 查询中，通过调用 SetOptions()函数为该查询中所有 Hadoop 作业优先级设置成 0：

```
setOptions({ conf: { "flex.priority": 0} });
```

还可以在命令行中启动一个 Hadoop 作业时指定优先级，比如：

```
hadoop jar $BIGINSIGHTS_HOME/IHC/hadoop*examples.jar -Dflex.priority=1 <input><output>
```

5.3.11 压缩

当使用 InfoSphere BigInsights 处理海量数据时，可以考虑使用数据压缩的方法减少对存储空间的需求以及加速 MapReduce 处理性能。IBM BigInsights 平台支持 Hadoop 开源框架中所有的压缩算法，并额外提供 IBM LZO 压缩(扩展名.cmx)，该压缩支持拆分压缩文件(通过 IBM LZO 压缩编译解码器)，保证 MapReduce 作业可以并行处理各压缩拆分部分。BigInsights 会根据文件扩展名自动识别解压缩文件所需的压缩算法，目前支持的

splittable 压缩如表 5.2 所示。

表 5.2 BigInsights 3.0 支持的 splittable 压缩算法

文件扩展名	用来解压缩文件的压缩算法
.cmx	IBM LZO
.bz2	BZip2
.gz	Gzip
.deflate	DEFLATE
.snappy	Snappy

5.3.12 总结

IBM 大数据平台战略可以帮助企业解决大数据的挑战，支持与客户现有的系统集成，具体包括信息整合、Hadoop 企业版系统（InfoSphere BigInsights）、流计算（InfoSphere Streams）、数据仓库（pureData Systems、InfoSphere Warehouse）、NoSQL 数据库（Cloudant）、加速器（Accelerator）、可视化与发现（InfoSphere Data Explorer）、应用程序开发、系统管理、数据治理和元数据管理等功能。大数据依赖于一个可缩放且可扩展的信息基础，只有企业的信息基础到位，并支持迅速增长的数据数量、多样性和速度，才能显著地从大数据中获益。

IBM 大数据平台提供全面的企业级 Hadoop 平台、流计算和分析引擎。企业级数据仓库实现各种数据的存储、分析和挖掘，结构化数据可以使用数据仓库进行存储和分析，结构化原始数据、半结构化和非结构化数据可以存储到企业级 Hadoop 系统中并进行后续分析处理，动态的半结构化和非结构化信息流通过流计算和分析引擎进行处理、过滤和分析后，生成的结构化分析结果可以存储到数据仓库中以便和其他历史信息进行进一步的深入分析，而获取的半结构化和非结构化信息可以存储到企业级 Hadoop 系统中（当然结构化的信息也可以存储到企业级 Hadoop 中）。数据仓库技术和企业级 Hadoop 系统之间是相互补充，而非相互竞争关系。借助于企业级 Hadoop 平台 InfoSphere BigInsights 3.0，用户可以方便地进行大数据分析、数据挖掘、文本分析、全文检索、报表展现以及各种交互式查询访问，使得企业用户进行大数据分析的时间价值曲线变平。



大数据环境下的安全与治理

6.1 大数据环境下的信息安全

在大数据时代，无处不在的智能终端、互动频繁的社交网络和超大容量的数字化存储，不得不承认大数据已经渗透到各个行业领域，逐渐成为一种生产要素发挥着重要作用，成为未来竞争的制高点。大数据所含信息量较高，虽然相对价值密度较低，但是对它里面所蕴藏的潜在信息，随着快速处理和分析提取技术的发展，可以快速捕捉到有价值的信息以提供参考决策。然而，大数据掀起新一轮生产率提高和消费者盈余浪潮的同时，随之而来的是信息安全的挑战。

6.1.1 信息安全的发展

如图 6.1 所示，20 世纪 80 年代，保证信息安全的层次是物理安全层次，使用的手段主要是机房管理、门禁管理等传统手段。在 20 世纪 90 年代，保证信息安全的层次是网络安全层次，使用的手段主要是防病毒和防火墙等手段。在 2000—2010 年，保证信息安全的层次是应用安全层次，使用的手段主要是身份验证、入侵防护、流量控制、VPN、内网安全等。在 2010 年之后，保证信息安全的层次是数据安全层次，使用的手段是数据防漏、数据防抵赖等。

6.1.2 数据安全的概念

国际标准化组织(ISO)对计算机系统安全的定义是：为数据处理系统建立和采用的技术与管理的安全保护，保护计算机硬件、软件和数据不因偶然和恶意的原因遭到破坏、更改和泄露。

数据安全有两方面的含义：一是数据本身的安全，主要是指采用现代密码算法对数据进行主动保护，如数据保密、数据完整性、双向身份认证等；二是数据防护的安全，主要是采用现代信息存储方法对数据进行主动防护，如通过磁盘阵列、数据备份、异地容灾等方法保证数据的安全。数据安全是主动的包含措施，数据本身的安全必须基于可靠的加密算法与

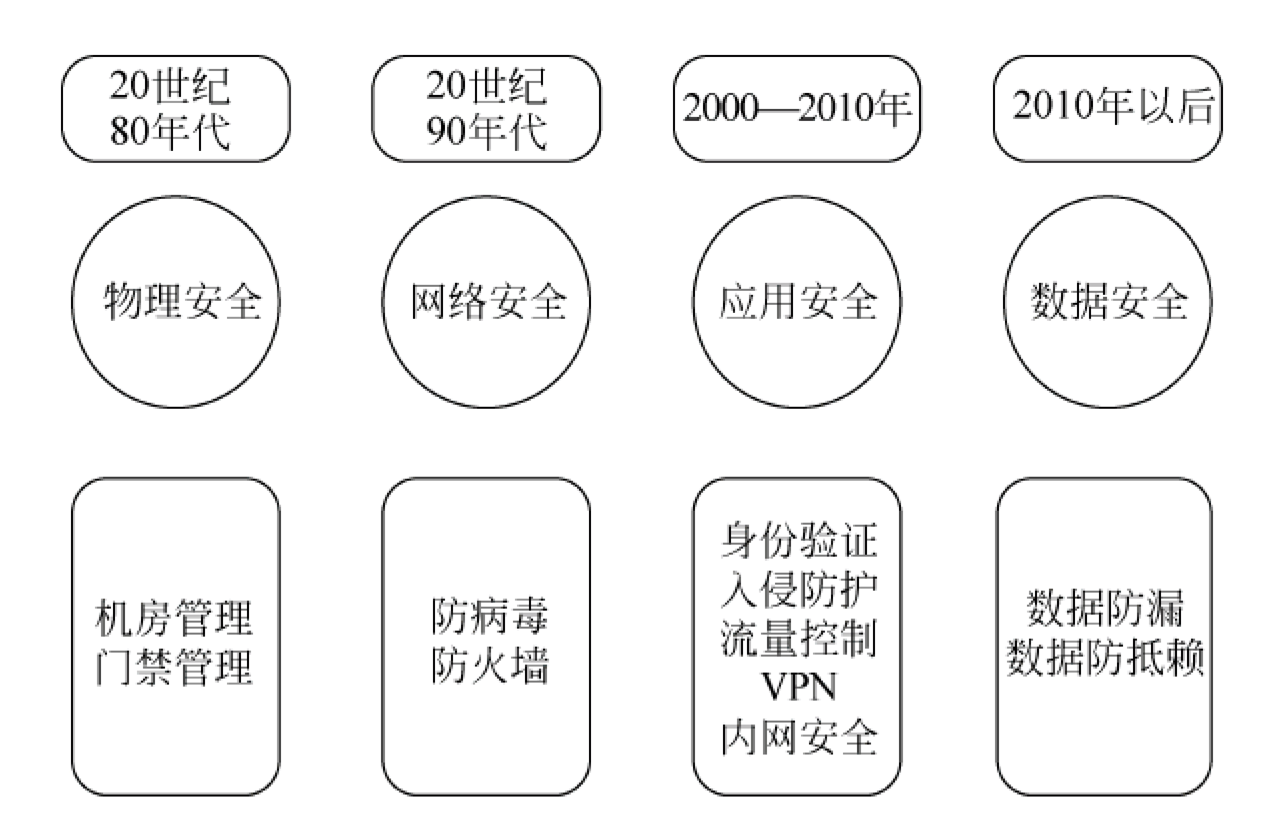


图 6.1　信息安全发展

安全体系，主要有对称算法与公开密钥密码体系两种。数据安全的基本特点包括：

- 机密性（Confidentiality）：机密性是指个人或团体的信息不为其他人获得。在计算机中，许多软件如邮件软件、网络浏览器等，都有保密性相关的设定，用以维护用户信息的保密性。
- 完整性（Integrity）：数据完整性是数据安全的 3 个基本要点之一，指在传输、存储信息或数据的过程中，确保信息或数据不被未授权的篡改或在篡改后能够被迅速发现。
- 可用性（Availability）：数据可用性是以使用者为中心的设计概念，易用性设计的重点在于让产品的设计能够符合使用者的习惯与需求。以互联网网站的设计为例，希望让使用者在浏览的过程中不会产生压力或感到挫折，并能让使用者在使用网站功能时，能用最少的努力发挥最大的效能。

6.1.3　大数据的特征

大数据通常被认为是一种数据量很大、数据形式多样化的非结构化数据。随着对大数据研究的进一步深入，大数据不仅指数据本身的规模，也包括数据采集工具、数据存储平台、数据分析系统和数据衍生价值等要素。其主要特点包括：

- 数据量大：大数据时代，各种传感器、移动设备、智能终端和网络社会等无时无刻都在产生数据，数量级别已经突破 TB，发展至 PB 乃至 ZB，统计数据量呈千倍级别上升。
- 类型多样：当今大数据不仅仅是数据量的飞速增长，而且还包含数据类型的多样化。以往数据大都以二维结构呈现，但随着互联网、多媒体等技术的快速发展和普及，视频、音频、图片、邮件、HTML、RFID、GPS 和传感器等产生的非结构化数据，每年都以 60%左右的速度增长。
- 运算高效：基于云计算的 Hadoop 大数据框架，利用集群的高速运算和存储，实现分布式运行系统，以流的形式提供高传输率来访问数据，适应了大数据的应用程序。而且，数据挖掘、语义引擎、可视化分析等技术的发展，可从海量的数据中深度解析，提取信息，是掌控数据增值的“加速器”。

- 产生价值：价值是大数据的终极目的。大数据本身可以说是一个"金矿产"，可以从大数据的融合中获得意想不到的有价值的信息。特别是激烈竞争的商业领域，数据正成为企业的新型资产，追求数据最大价值化。同时，大数据价值也存在密度低的特性，需要对海量的数据进行挖掘分析才能得到真正有用的信息，形成用户价值。

6.1.4　大数据给信息安全带来新的挑战和机遇

（1）大数据包含隐私信息，加大泄露风险

网络空间中的数据来源涵盖非常广阔的范围，大量数据的聚集不可避免地加大了用户隐私泄露的风险。一方面，大量的数据汇集，包括大量的企业运营数据、客户信息、个人的隐私和各种行为的细节记录。这些数据的集中存储增加了数据泄露风险。另一方面，一些敏感数据的所有权和使用权并没有明确的界定，很多基于大数据的分析都未考虑到其中涉及的个体的隐私问题。

（2）大数据存储带来新的安全挑战

数据大集中的后果是复杂多样的数据存储在一起，例如开发数据、客户资料和经营数据存储在一起，可能会出现违规地将某些生产数据放在经营数据存储位置的情况，造成企业安全管理不合规。大数据的大小影响到安全控制措施能否正确运行。对于海量数据，常规的安全扫描手段需要耗费过多的时间，已经无法满足安全需求。安全防护手段的更新升级速度无法跟上数据量非线性增长的步伐，大数据安全防护面临巨大挑战。

（3）大数据越来越成为"攻击"的目标

大数据越来越成为被关注的目标。因为一方面，大数据不仅意味着海量的数据，也意味着更复杂、更敏感的数据，这些数据会吸引更多的潜在攻击者，成为更具吸引力的目标。另一方面，数据的大量聚集，使得黑客一次成功的攻击能够获得更多的数据，无形中增加了攻击的"收益率"。

（4）大数据使传统的防护检测难以有效进行

传统的检测是基于单个时间点进行的基于威胁特征的实时匹配检测，而高级可持续攻击（APT）是一个实施过程，并不具有能够被实时检测出来的明显特征，无法被实时检测。同时，APT 攻击代码隐藏在大量数据中，很难被发现。此外，大数据的价值低密度性，让安全分析工具很难聚焦在价值点上，黑客可以将攻击隐藏在大数据中，给安全服务提供商的分析制造很大困难。

（5）大数据技术的价值和风险是并存的

在企业使用大数据技术获取商业价值的同时，黑客也在利用这些大数据技术向企业发起攻击。黑客最大限度地收集更多有用信息，大数据为黑客发起攻击提供了更多机会，大数据分析让黑客的攻击更精准。

（6）大数据技术为信息安全发展提供了新机遇

大数据正在为安全分析提供新的可能性，对于海量数据的分析有助于信息安全服务提供商更好地刻画出网络异常行为，从而找出数据中的风险点。对实时安全和商务数据结合在一起的数据进行预防性的分析，以便识别钓鱼攻击，防止诈骗和阻止黑客入侵。网络攻击行为总会留下蛛丝马迹，这些痕迹都以数据的形势隐藏在大数据中，利用大数据技术整合计算和处理资源有助于更有针对性的应对信息安全威胁，使得网络攻击行为无所遁形，有助于

找到发起攻击的源头。

6.2 大数据面临的安全威胁

相对于传统数据的安全保护,大数据的安全保护更加复杂。一方面,大数据中包括大量的企业运营数据、客户信息、个人的隐私和各种行为的细节记录,这些数据的存储增加了数据泄露的风险,使大数据安全面临更多的威胁。另一方面,大数据给数据完整性、可用性和保密性带来了更多挑战,传统的安全工具已不再像以前那么有效。

大数据的特征为黑客发起攻击提供了更多机会,利用大数据,黑客可以扩大攻击的效果。如黑客利用大数据发起网络攻击,这个数量级是传统单点攻击所不具备的。其次,黑客可以通过控制关键节点放大攻击效果。此外,大数据的低密度性使得现有的安全分析工具很难聚焦在关键点上,黑客可以将攻击隐藏在大数据中,给安全厂商的分析制造困难。因此,大数据的安全保护面临着更多威胁。

(1) 基本威胁

- 信息泄露:信息泄露主要指大数据中的部分或全部信息被透露给未被授权的用户、软件或实体,尤其是大数据中一些敏感信息或关键信息的泄露。这种威胁通常来自窃听、搭线或其他更加复杂的信息探测攻击。
- 完整性破坏:由于非授权的增加、删除和修改等操作,大数据中的部分信息丢失,完整性遭到破坏。
- 拒绝服务:用户对大数据中一些资源的合理访问被无条件地拒绝。主要包括两种情况:①攻击者制造一系列非法的访问,致使系统产生过量负荷,从而导致系统资源在合法用户看来是无法使用的;②大数据处理系统在物理上或逻辑上遭到破坏,致使用户的合理请求服务被拒绝。
- 非法使用:非法使用主要指大数据中的一些资源被非授权用户使用,或者是大数据中的某些资源被用户以非授权的方式使用。

(2) 恶意威胁

在大数据的安全防护中,恶意威胁应该引起高度重视,因为它有可能导致其他威胁的相继发生。恶意威胁主要分为渗入威胁和植入威胁两类。

主要的渗入威胁分为 3 类:

- 假冒:系统中某个实体假装成另外一个不同的实体,以方便获取系统的权限和特权。该攻击方法较为常见,黑客多采用这种方式实施攻击。
- 旁路控制:为了获取非授权的权利和特权,攻击者会寻找系统自身的缺陷和漏洞,绕过系统的安全防线对大数据实施攻击。
- 授权侵犯:一个授权的用户将其权限用于其他非授权的目的,这种攻击又称为“内部攻击”。

主要的植入威胁可分为两类:

- 特洛伊木马:主要指软件中含有用户觉察不出的程序段,当该程序段被执行时,用户数据的安全性会遭到破坏。
- 陷阱门:主要指一些用户或程序在大数据管理系统的某个或多个部件中设置“机

关”，当大数据系统接收到特定的输入信息时，允许违反正常的安全策略。例如，当一个用户登录大数据管理的子系统时，若系统设置有陷阱门，攻击者输入一个特定的用户身份时，便可以绕过正常的口令检测或身份认证过程而直接侵入系统内部。

(3) 潜在威胁

在大数据的安全防护过程中，潜在威胁有可能导致一些更基本威胁的发生。常见的潜在威胁分为4种类型：

- 窃听(Eavesdropping)；
- 流量分析(Traffic Analysis)；
- 操作人员的不慎所导致的信息泄露；
- 媒体废弃物所导致的信息泄露。

图6.2给出了大数据面临的一些典型威胁及其相互关系。图中的路径可以交错，如假冒攻击可以成为所有基本威胁的基础，同时假冒攻击本身也存在信息泄露的潜在威胁。

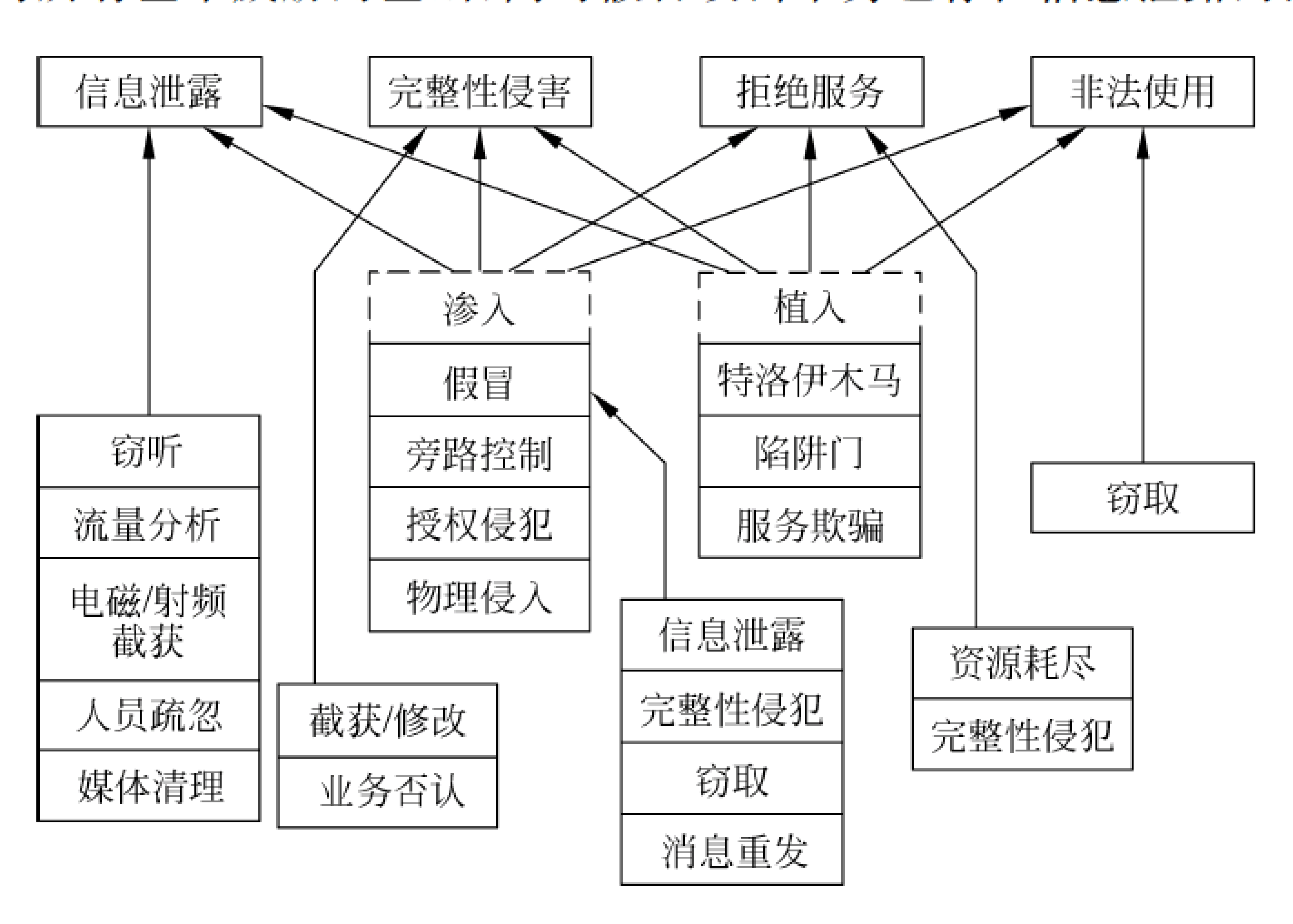


图6.2　典型的威胁及其相互关系

6.3　大数据安全的应对策略

当然，大数据也为数据安全的发展提供了新机遇，为安全分析提供新的可能性，对海量数据的分析有助于更好地跟踪网络异常行为，对实时安全和应用数据结合在一起的数据进行预防性分析，可防止诈骗和黑客入侵。网络攻击行为总会留下蛛丝马迹，这些痕迹都以数据的形式隐藏在大数据中，从大数据的存储、应用和管理等方面层层把关，可以有针对性地应对数据安全威胁。

6.3.1　大数据存储安全策略

基于云计算架构的大数据，数据的存储和操作都是以服务的形式提供。目前，大数据的安全存储采用虚拟化海量存储技术来存储数据资源，涉及数据传输、隔离、恢复等的问题。通过以下几种方法解决大数据的安全存储：

（1）数据加密。在大数据安全服务的设计中，大数据可以按照数据安全存储的需求，存储在数据集的任何存储空间，通过 SSL（安全套接层）加密，实现数据集的节点和应用程序之间移动保护大数据。在大数据的传输服务过程中，加密为数据流的上传与下载提供有效的保护。应用隐私保护和外包数据计算，屏蔽网络攻击。目前，PGP 和 TrueCrypt 等程序都提供了强大的加密功能。

（2）分离密钥和加密数据。使用加密把数据使用与数据保管分离，把密钥与要保护的数据隔离开。同时，定义产生、存储、备份、恢复等密钥管理生命周期。

（3）使用过滤器。通过过滤器的监控，一旦发现数据离开了用户的网络，就自动阻止数据的再次传输。

（4）数据备份。通过系统容灾、敏感信息集中管控和数据管理等产品，实现端对端的数据保护，确保大数据损坏情况下有备无患和安全管控。

6.3.2 大数据应用安全策略

随着大数据应用所需技术和工具的快速发展，大数据应用安全策略主要从以下几方面着手：

（1）防止 APT 攻击。借助大数据处理技术，针对 APT 安全攻击隐蔽能力强、长期潜伏、攻击路径和渠道不确定等特征，设计具备实时检测能力与事后回溯能力的全流量审计方案，提醒隐藏有病毒的应用程序。

（2）用户访问控制。大数据的跨平台传输应用在一定程度上会带来内在风险，可以根据大数据的密集程度和用户需求的不同，将大数据和用户设定不同的权限等级，并严格控制访问权限。而且，通过单点登录的统一身份认证与权限控制技术，对用户访问进行严格的控制，有效地保证大数据应用安全。

（3）整合工具和流程。通过整合工具和流程，确保大数据应用安全处于大数据系统的顶端。整合点平行于现有连接的同时，减少通过连接企业或业务线的 SIEM 工具输出到大数据安全仓库，以防止预处理的数据暴露以及加工后的数据集溢出。通过设计标准化的数据格式简化整合过程，也可以改善分析算法的持续验证。

（4）数据实时分析引擎。数据实时分析引擎融合云计算、机器学习、语义分析、统计学等多个领域，从大数据中第一时间挖掘出黑客攻击、非法操作、潜在威胁等各类安全事件，发出警告响应。

6.3.3 大数据管理安全策略

云计算专家说："数据安全三分靠技术，七分靠管理"。通过技术来保护大数据的安全必然重要，但管理也很关键。大数据的管理安全策略主要包括：

（1）规范建设。大数据建设是有序的、动态的、可持续发展的系统工程，建设规范的运行机制、标准和共享平台至关重要。规范化建设可以促进大数据管理过程的正规有序，实现各级各类信息系统的网络互连、数据集成、资源共享，在统一的安全规范框架下运行。

（2）建立以数据为中心的安全系统。基于云计算的大数据存储在云共享环境中，为了大数据的所有者可以对大数据使用进行控制，可以通过建设一个基于异构数据为中心的安全方法，从系统管理上保证大数据的安全。

(3) 融合创新。大数据是在云计算的基础上提出的新概念,大数据时代应以智慧创新理念融合大数据与云计算,以智能管道与聚合平台为基础,提升数据流量规模、层次及内涵,在大数据流中提升知识价值洞察力。积极创造大数据公司技术融合平台,寻找数据洪流大潮中新的立足点,特别是在数据挖掘、人工智能、机器学习等新技术的创新应用中融合创新。

6.4 大数据的治理

数据治理涵盖了整个企业的信息化立场和建设,包括所有核心业务系统、运营数据存储或数据仓库、ECIF、风险控制以及很多其他的系统,需要技术和"软内容"平台提供支撑,包括主数据管理平台、数据质量管理平台、数据服务平台以及元数据管理和数据标准管理,最终实现所有数据的全方位监管,实现端到端的数据梳理和管理,确保数据的有效性、可访问性、高质量、一致性、可审计和安全性,这是数据治理的目标所在。

6.4.1 大数据环境下的安全技术体系框架

在大数据环境下的安全技术体系框架比较复杂,有各种终端、防护方法等。图 6.3 展示了大数据环境下的安全技术体系框架。

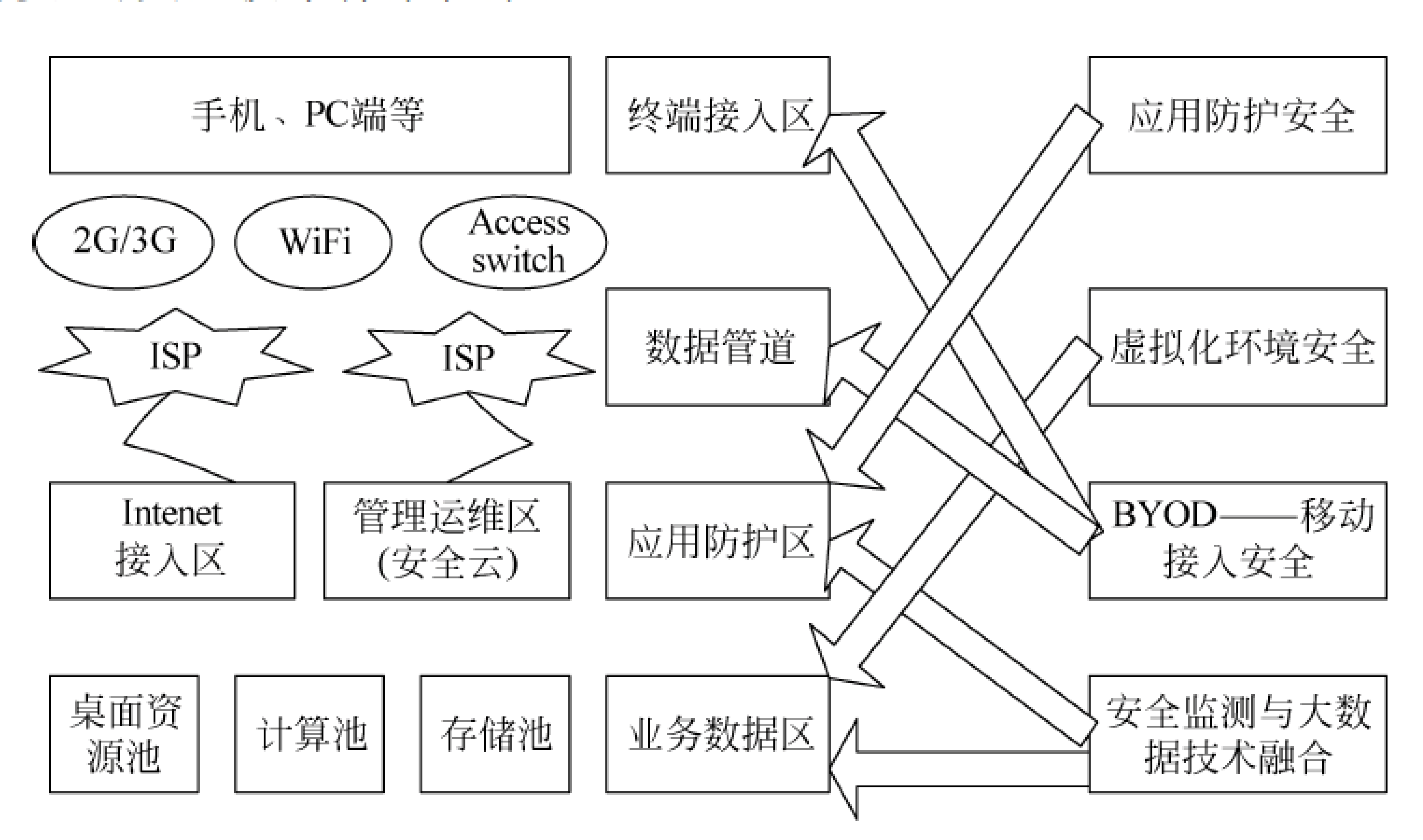


图 6.3　大数据环境下的安全技术体系框架

6.4.2 大数据治理定义

数据治理是指在企业数据生命的整个周期(从数据采集到数据使用直至数据存档)制定由业务推动的数据政策、数据所有权、数据监控、数据标准以及指导方针。数据治理的重点在于,要将数据明确作为企业的资产看待。数据治理的六大支柱如图 6.4 所示,对于任何企业数据治理计划而言,变更管理都是必不可少的核心部分。

6.4.3 数据治理的作用

数据治理在架起 IT 技术与业务之间的桥梁方面发挥作用,如图 6.5 所示。

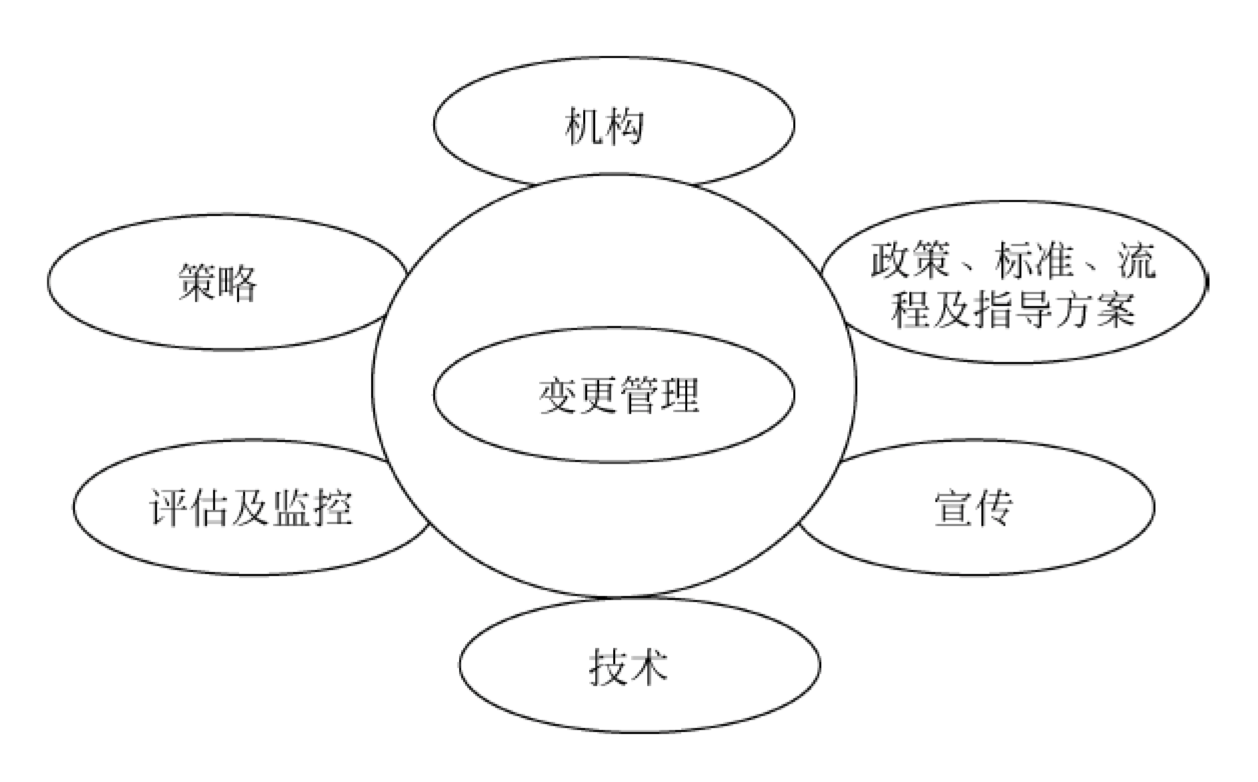


图 6.4 数据治理六大支柱

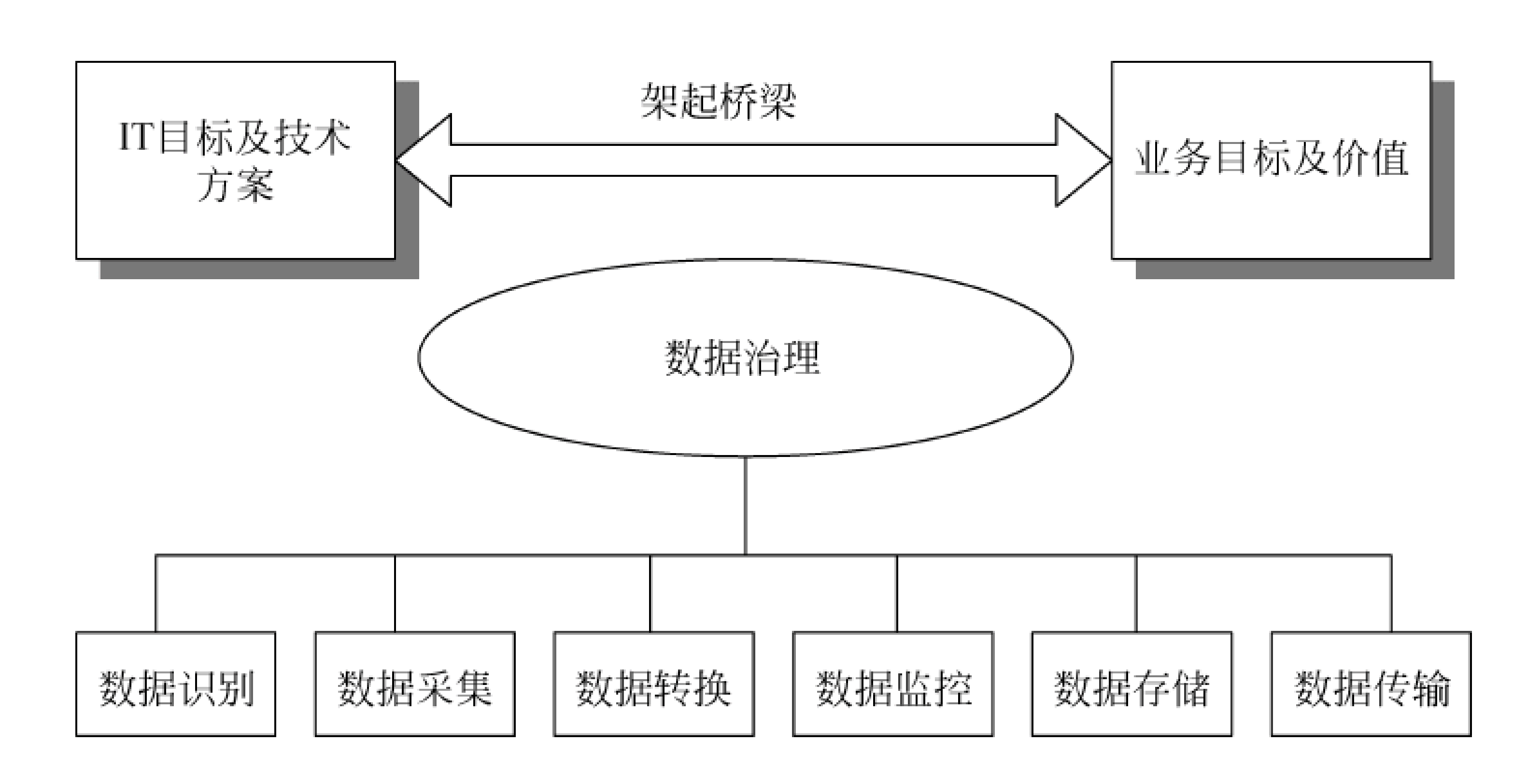


图 6.5 数据治理在业务目标与 IT 目标之间架起桥梁

6.5 大数据加密技术

由于大数据的加密处理对存储资源、计算资源等都提出了较高的性能要求，传统加密算法和处理工具已不再像以前那么高效。目前，针对大数据加密的处理，常见方式主要有两种：

（1）数据采样。为了使数据加密处理更有针对性，通过数据采样的方法，搜集关键信息的数据域，客观上会将数据规模变小，以加快处理速度。然而在某些领域，应用采样技术仅对大数据中的部分信息进行加密，很难防范攻击者从其他信息中挖掘出关键信息和敏感信息的可能性。

（2）分而治之。利用分布式计算技术，将大数据分别在不同的计算机上进行加密处理，从而有效提高了大数据加解密的速度。常见的是使用 MapReduce 模式对海量数据进行并行处理。

以上两种加密技术的通用加密模型是一致的，如图 6.6 所示。

大数据加密系统可表示为一个五元组：<P,C,K,E,D>，其中

- P 表示交互数据集合。
- C 表示加密数据集合。

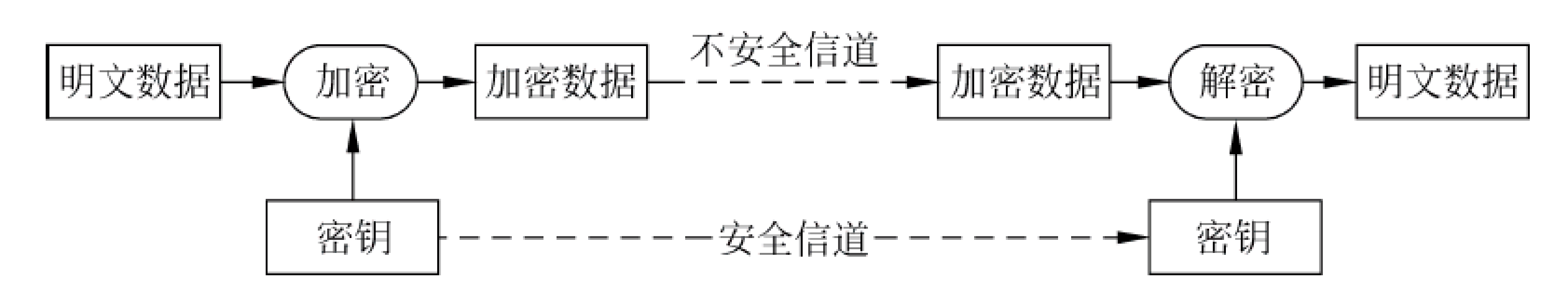


图 6.6　大数据加密通用模型

- K 表示密钥集合。
- E 表示加密函数：利用设计的加密函数 E 和密钥 k(k∈K)对交互数据 p(p∈P)进行加密，得加密数据 c(c∈C)，即 E:P×K→C，简记为 E_k(P)=C。
- D 表示解密函数：利用设计的解密函数 D 和密钥 k(k∈K)对加密数据 c 进行解密，得解密数据 p，即 D:C×K→P，简记为 D_k(C)=P。

在大数据加密方面，国内外研究已取得了初步成果，主要包括 4 个方面：

(1) 基于现代密码体制的大数据加密技术

- 加密原理：1949 年，Shannon 发表了“保密系统的通信理论”一文，对信息保密问题做了全新的诠释，从此逐渐形成了现代密码学理论。现代密码体制可分为对称密码体制和非对称密码体制。对称密码体制中，通信双方共享一个加密密钥和一个解密密钥，加、解密密钥相同或彼此容易相互确定，其代表为数据加密标准(Data Encryption Standard，DES)和高级加密标准(Advanced Encryption Standard，AES)。非对称密码体制中，发送方拥有一个公钥(可公开)和一个私钥(必须保密)，其代表为 RSA(Rivest，Shamir，Adleman)和椭圆曲线密码体制(Elliptic Curve Cryptosystem，ECC)。对椭圆曲线离散对数密码体制(ECDLC)安全性的研究已进行了十余年，尚未发现明显的弱点。在实际应用中，对称密码体制主要用于加密文本信息。非对称密码体制比对称密码体制平均慢约 1000 倍，它经常用来加密短信息，如密钥等。基于现代密码体制的大数据加密原理是将大数据视为二进制流，采用现代密码体制进行加解密。
- 技术特征：理论上，大数据完全可用现代密码体制进行加密，但是大数据是特殊的数据(数据量大、高维数和冗余度高等)，而现代密码体制多是针对文本数据进行加密设计的，并没有结合大数据加密的安全需求。且往往结构复杂、计算量大，因此很难满足实际应用需求。单纯依靠增加加解密设备、存储设备和带宽并不能从根本上解决问题，必须寻求新的技术解决途径。

(2) 基于生物工程的大数据加密技术

- 加密原理：生物工程中 DNA 计算是近年来出现的一种新型自然计算方法，因其高度的并行性和海量的存储能力，引起了广泛关注。1994 年，Adleman 首次提出了 DNA 计算，开创了信息处理的新阶段。目前，DNA 加密已成为国际密码学研究的前沿领域。基于 DNA 计算的大数据加密算法是一种新颖的数据加密手段，以 DNA 为信息载体，以现代生物技术为实现工具，挖掘 DNA 固有的高存储密度和高并行性等优点，实现加密认证及签名等密码学功能，具有较好的研究前景。

张强教授已建立了基于图像的 DNA 编码和运算理论，提出了多种基于 DNA 的图像加密算法。同时，还定义了 DNA 编码的加法和减法运算。

- 技术特征：张强教授设计的基于 DNA 计算的加密算法，是建立在 A、G、C 和 T 4 种编码与图像像素之间的一种转换关系上的，并非通过真正的 DNA 生物技术实现，此类编码加法和减法运算规则有限，自身安全性不高，需要结合其他加密手段来进一步提高安全性。

目前，基于 DNA 计算技术的大数据加密方案还处于研究初期，但 DNA 分子固有的超大规模并行性、超低的能量消耗和超高的存储密度，使得基于 DNA 计算技术的大数据加密方案具有传统密码算法所不具有的独特优势。

（3）基于属性基的大数据加密技术

- 加密原理：Shamir 于 1979 年首次提出身份基加密的概念，随后众多的身份基加密的扩展概念被提出，比如层次化身份基加密、身份基广播加密、空间加密等。属性基加密也是身份基加密的一种扩展。目前正在使用两种属性基加密方式分别为密钥政策基加密和密文政策基加密。在密文政策基属性基加密处理中，明文消息在一个可表示成由“与”、“或”、“非”门连接的逻辑表达式的谓词下进行加密。每个用户从属性机构处获得对应某个属性集合的密钥。当且仅当该属性集合满足上述谓词时，解密才能成功。密钥政策基属性基加密的情况则与此相反：在该加密体制中，密文和属性集合相对应，而用户密钥则对应某个谓词。在实际应用中，可根据用户属性及对数据的保密级别，对大数据进行不同层次的加密处理，此方法在敏感大数据的机密性保护中具有较好的应用。
- 加密特征：基于属性基的加密机制具有以下 4 个常见特点：

① 资源提供方仅根据属性信息加密消息，不需要关注群体中的成员数量和身份特性，降低了数据加解密开销，同时保护了用户的隐私。

② 只有符合密文属性要求的成员才能解密消息，从而保证了数据的机密性。

③ 基于属性基的加密机制中，用户密钥由随机的多项式或者随机数产生，保证了密钥的新鲜性，各个不同的用户无法将其密钥进行联合，从而有效阻止了串谋攻击。

④ 支持基于属性的灵活访问控制策略，可以灵活地实现属性的与、或、非及门限的操作，处理起来非常高效。

基于属性基的大数据加密机制的高效性、抗串谋性和策略表示的灵活性，使其在大数据的细粒度访问控制、信息的定向广播、组密钥管理及隐私保护等领域具有良好的应用前景。

（4）基于并行计算的大数据加密技术

- 加密原理：近 20 年来，单机尤其是 CPU、内存等硬件的性能有了较大提高。然而，硬件技术的发展在理论上是非常有限的。如果说硬件的发展在纵向上提高了系统加解密的性能，那并行技术的发展则是从横向上拓展了数据加解密处理的方式。并行处理采用的策略是“分而治之”，从性能延伸的意义上来看，该方法在理论上对于系统性能的延伸是没有边界的。

有学者将 IBE 与 CPK 有机结合，能够为超大规模的云计算用户提供安全支持，具有大规模、高效率、低带宽、易使用的特点。宋怀明等提出了一种“大规模数据密集型系统中的去重查询优化”算法，针对 shared-nothing 结构下大规模数据密集型系统的查询，提出了一种有效的数据分布策略和并行处理方法，分别对相关属性和无关属性去冗进行优化，为大数据加密过程中的去重查询提供了有效算法。韩希先等提出了一种新的大数据上的 top-k 查询

算法 TKEP，该算法在查询的增长阶段就执行剪切，从而大大减少增长阶段需要维护的候选元组，为大数据加密后的快速搜索提供了重要方法。

在实际应用中，比较主流的两种分布式数据处理技术是 MapReduce 技术和并行数据库技术。大数据可以通过 MapReduce 这一并行处理技术来提高数据的处理速度。MapReduce 的设计初衷是通过大量廉价服务器实现大数据并行处理，对数据一致性要求不高，其突出优势是具有扩展性和可用性，特别适用于海量的结构化、半结构化及非结构化数据的混合处理。

- 技术特征：空间或时间上的并行处理能够大大提高任务的处理效率。并行计算通常分为“任务并行处理”和“数据并行处理”两方面。任务并行处理会使任务的协调和管理变得非常复杂，不利于操作。数据并行处理则是将一个大任务分解为多个子任务，比任务并行处理容易。云计算是基于互联网的、典型的分布式计算方式，使超级计算能力通过互联网自由流通成为可能，它将巨大的分层资源池连接在一起，以提供各种 IT 服务。

大数据主要特点是存储密度大，计算强度大，需要大规模的并行存储和处理的能力。基于并行计算的大数据加密技术目前还是在起始阶段，但其超强的计算能力和快速处理能力在大数据加密处理时已表现出优越的性能，具有传统加密算法无法比拟的优势。

总之，大数据的加密技术总体沿着如何进一步提高安全性、加密速度、节约耗能，以及易扩展性、时效性和交互性的方向发展，未来可能的几个研究方向为：

(1) 大数据加密技术与分布式计算结合。在云计算等分布式环境下，如何处理大数据流的实时加解密，设计出适用于分布式环境下实时的加密算法有待研究。

(2) 大数据加密技术与搜索技术结合。大数据加密状态下如何解决密文信息的快速检索，有待研究与开发。

(3) 大数据加密技术与数据挖掘技术结合。大数据通常具有数据量大、高维和冗余度高的特点，如何在保证用户的基本需求下，在大数据密文中挖掘出有效信息还有待研究。

(4) 大数据加密自恢复能力验证。在大数据密文的传输过程中，存在被第三方篡改或网络丢包的可能性，在部分数据丢失的情况下，确保加密数据的自恢复能力，对大数据的完整性和可信性验证也十分必要。

(5) 大数据加密算法与数据消冗技术的结合。大数据具有类型数据多样化和冗余度高的特征，研究发现，应用系统保存的大数据中冗余度高达 60%，而且随着时间的推移冗余度不断增大，因此大数据的加密算法中数据消冗技术的研究具有重要意义和价值。



Hadoop技术

7.1 Hadoop 简介

7.1.1 简介

Apache Hadoop 是一款支持数据密集型分布式应用并以 Apache 2.0 许可协议发布的开源软件框架。它支持运行在商品硬件构建的大型集群上的应用程序。Hadoop 是根据 Google 公司发表的 MapReduce 和 Google 档案系统的论文实现而成。

Hadoop 框架透明地为应用提供可靠性和数据移动，实现了名为 MapReduce 的编程范式：应用程序被分割成许多小部分，而每个部分都能在集群中的任意节点上执行或重新执行。此外，Hadoop 还提供了分布式文件系统，用以存储所有计算节点的数据，这为整个集群带来了非常高的带宽。MapReduce 和分布式文件系统的设计，使得整个框架能够自动处理节点故障。Hadoop 使应用程序与成千上万的独立计算的计算机可以处理 PB 级的数据。现在普遍认为整个 Apache Hadoop"平台"包括 Hadoop 内核、MapReduce、Hadoop 分布式文件系统（HDFS）以及一些相关项目，有 Apache Hive 和 Apache HBase 等。

7.1.2 Hadoop 用途

Hadoop 的主要用途有如下几方面：

- 日志分析；
- 广告优化；
- 对大数据进行处理；
- 进行科学研究；
- 搜索引擎应用。

7.2　Hadoop 安装与简单配置

由于 Hadoop 需要运行在 Linux 中，如果计算机中没有 Linux 系统，要下载 Liunx 系统，并安装最新版的 JDK，再进行 Hadoop 安装和配置，本节将简要地讲解 Hadoop 单机模式的配置，如果对其他配置和使用方式有兴趣，可以到互联网上寻找资料。

7.2.1　Linux 安装

这里讲到的 Linux 系统采用 Cent OS(Community Enterprise Operating System)，Cent OS 为免费的 Linux 操作系统，它是 Linux 发行版之一，来自于 Red Hat Enterprise Linux 依照开放源代码规定释出的源代码所编译而成。

如果不想在计算机中进行实体 Linux 操作系统安装，可以采用虚拟机安装模式。

在首页选择右侧的 Downloads，在如图 7.1 所示页面，选择当前需要安装系统的计算机目前的操作系统所对应的版本进行下载。



图 7.1　Virtual Box 下载

在下载后，运行安装文件，并按照提示进行安装即可完成安装，如图 7.2 所示(这里采用 Apple OS X 系统，安装界面和使用可能与 Windows 操作系统有少许不同，但是按照提示即可完成安装和使用)。

安装后即可运行程序，进入 Virtual Box 程序界面，如图 7.3 所示。

下载 Cent OS 的安装镜像，访问 Cent OS 官网 http://www.centos.org/download/进行下载，如图 7.4 所示。

选择 DVD ISO，系统会提供一些下载地址，选择一个进行下载，如图 7.5 所示。

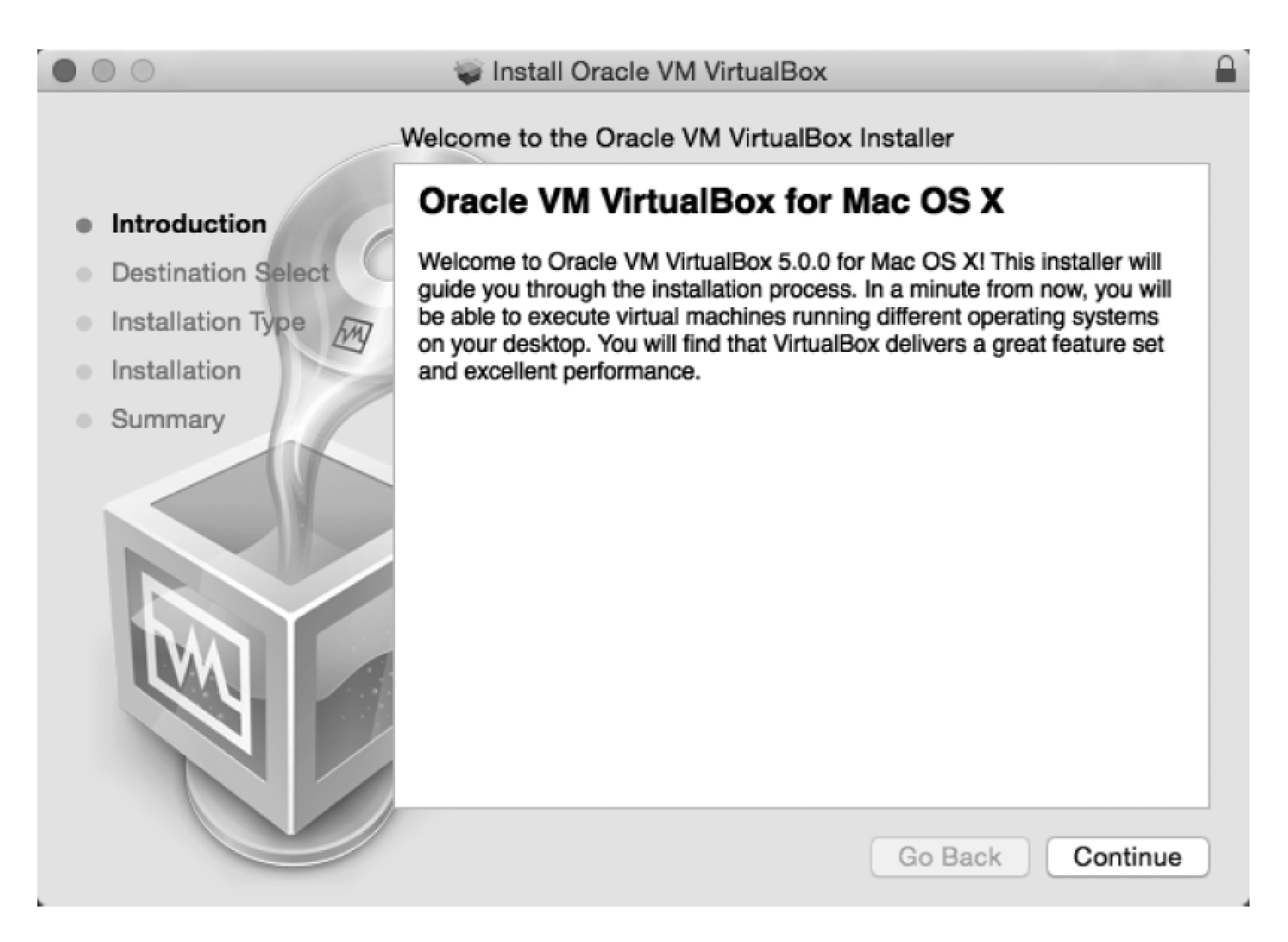


图 7.2　Virtual Box 安装界面



图 7.3　Virtual Box 界面

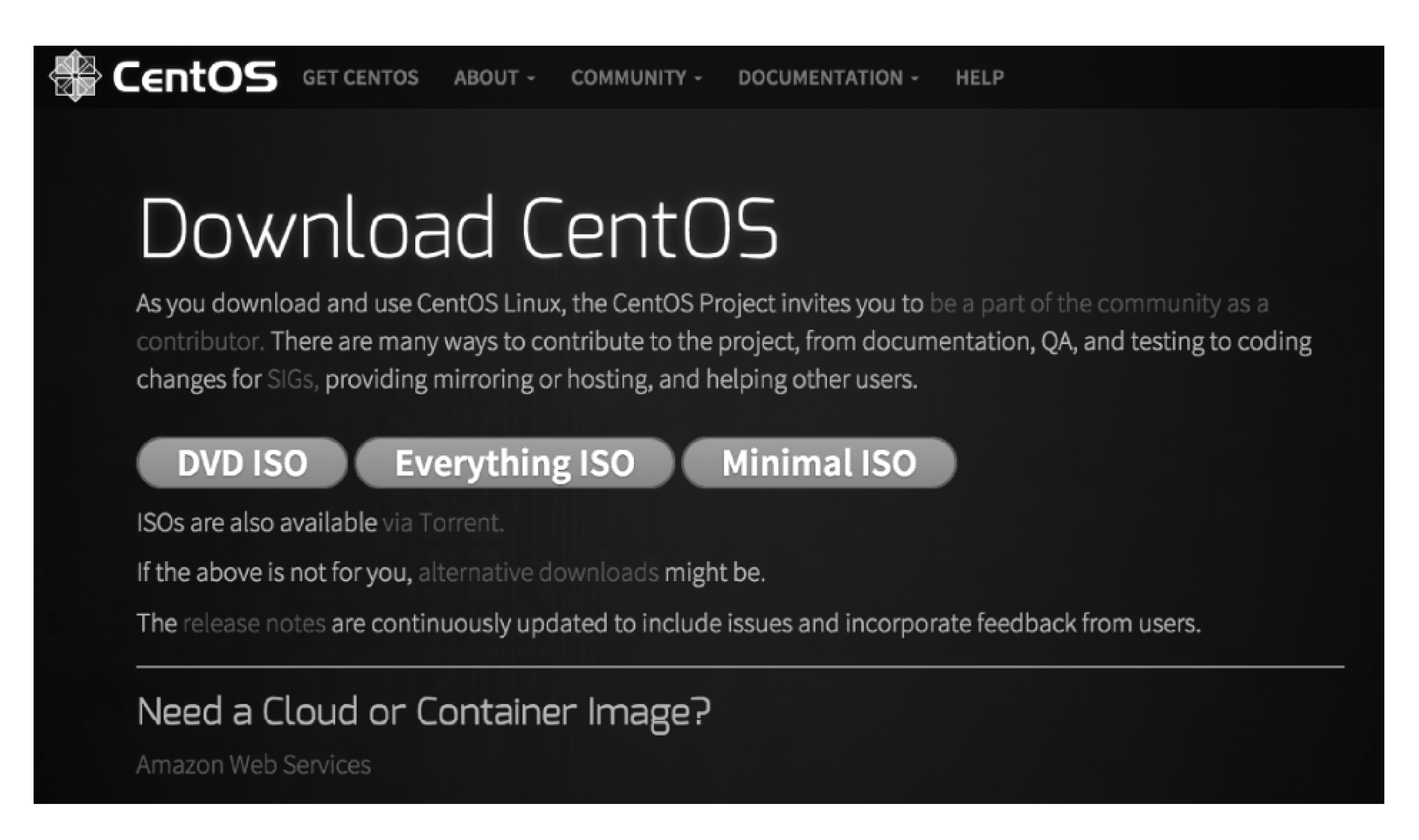


图 7.4　Cent OS 官方网站

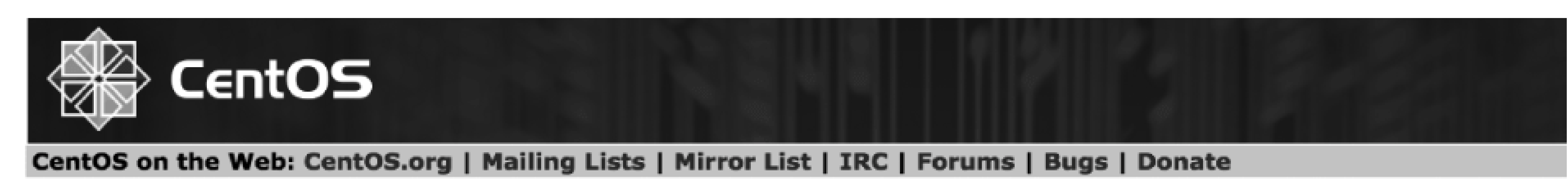


图 7.5　镜像下载地址选择

在选择之后等待下载完成。完成下载之后，打开 Virtual Box 软件，选择 New 命令。如图 7.6 所示，Name 设置为 Cent OS，Type 设置为 Linux，Version 设置为 Other Linux (64-bit)。

之后单击 Continue 按钮，进行内存设置，默认设置即可，如图 7.7 所示。

单击 Continue 按钮，进行硬盘设置，选择 Create a virtual hard disk now 选项，如图 7.8 所示。

单击 Create 按钮，进入硬盘文件类型选择界面。选择 VDI 模式即可，如图 7.9 所示。

单击 Continue 按钮，进行物理硬盘上的存储设置，选择 Dynamically allocated 即动态分配硬盘空间，如图 7.10 所示。

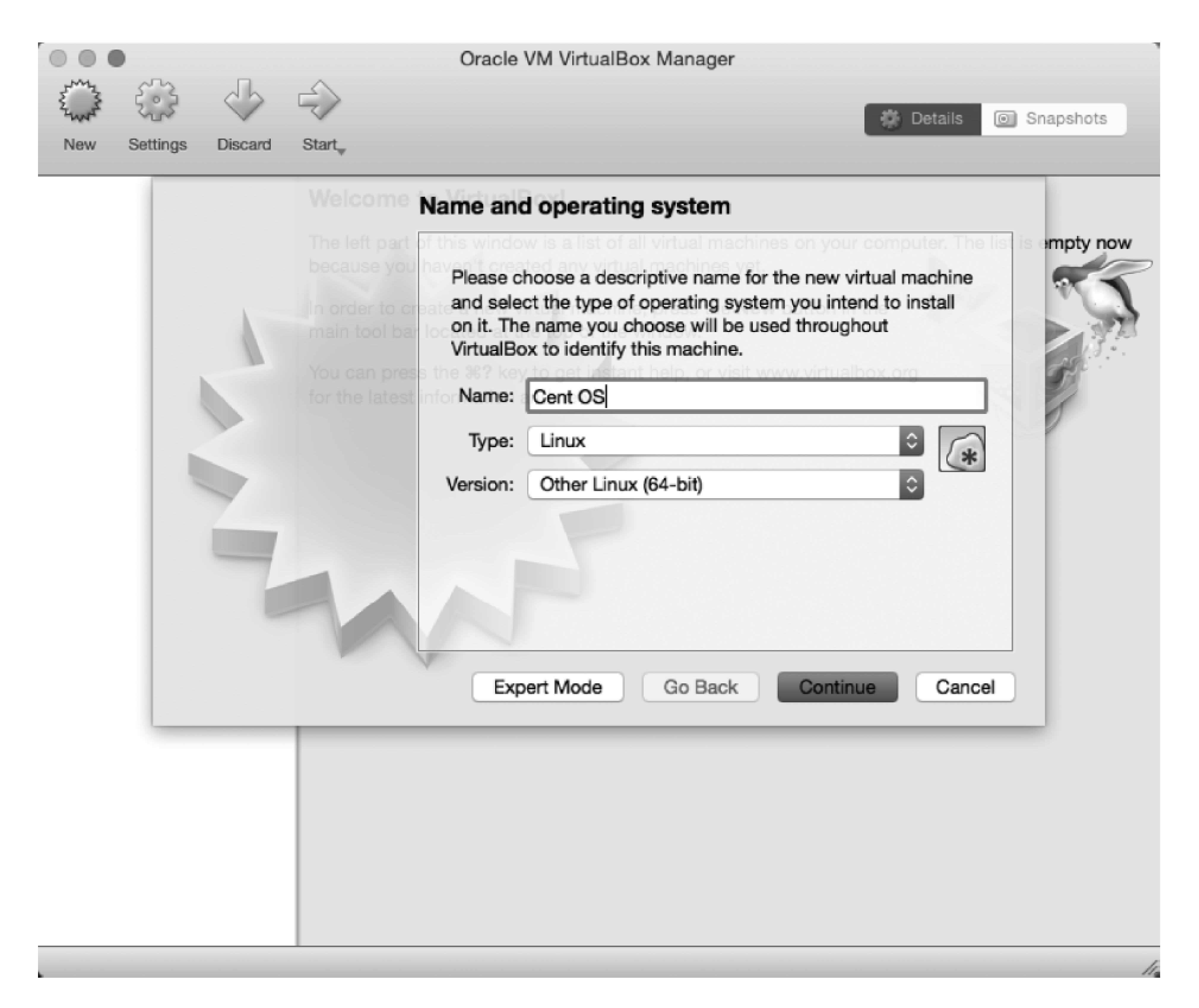


图 7.6 安装 Cent OS

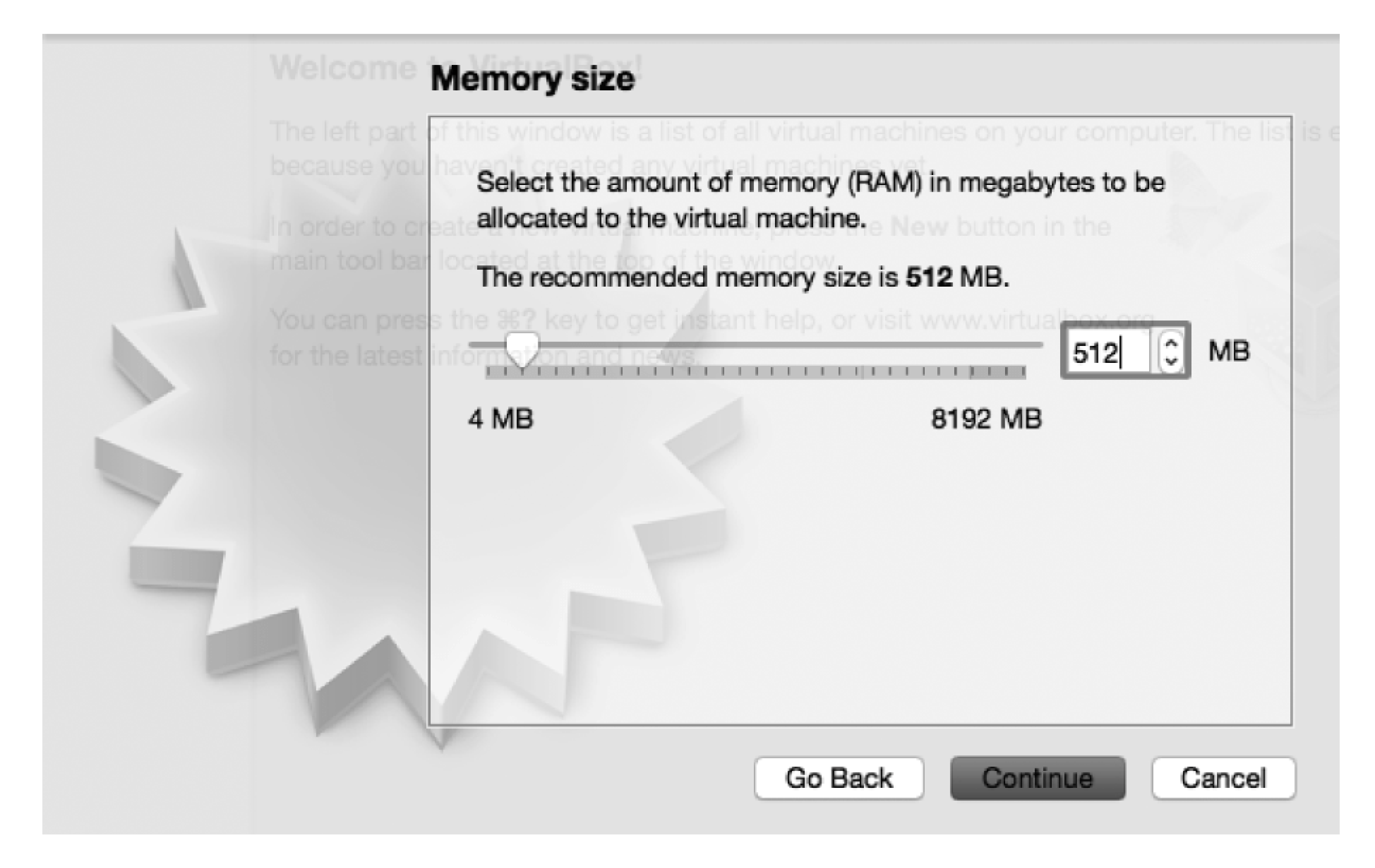


图 7.7 内存设置



图 7.8　硬盘设置



图 7.9　硬盘文件类型选择



图 7.10 物理硬盘上的存储设置

单击 Continue 按钮,进行硬盘文件地址设置和文件大小设置,设置为 10GB 即可,如图 7.11 所示。

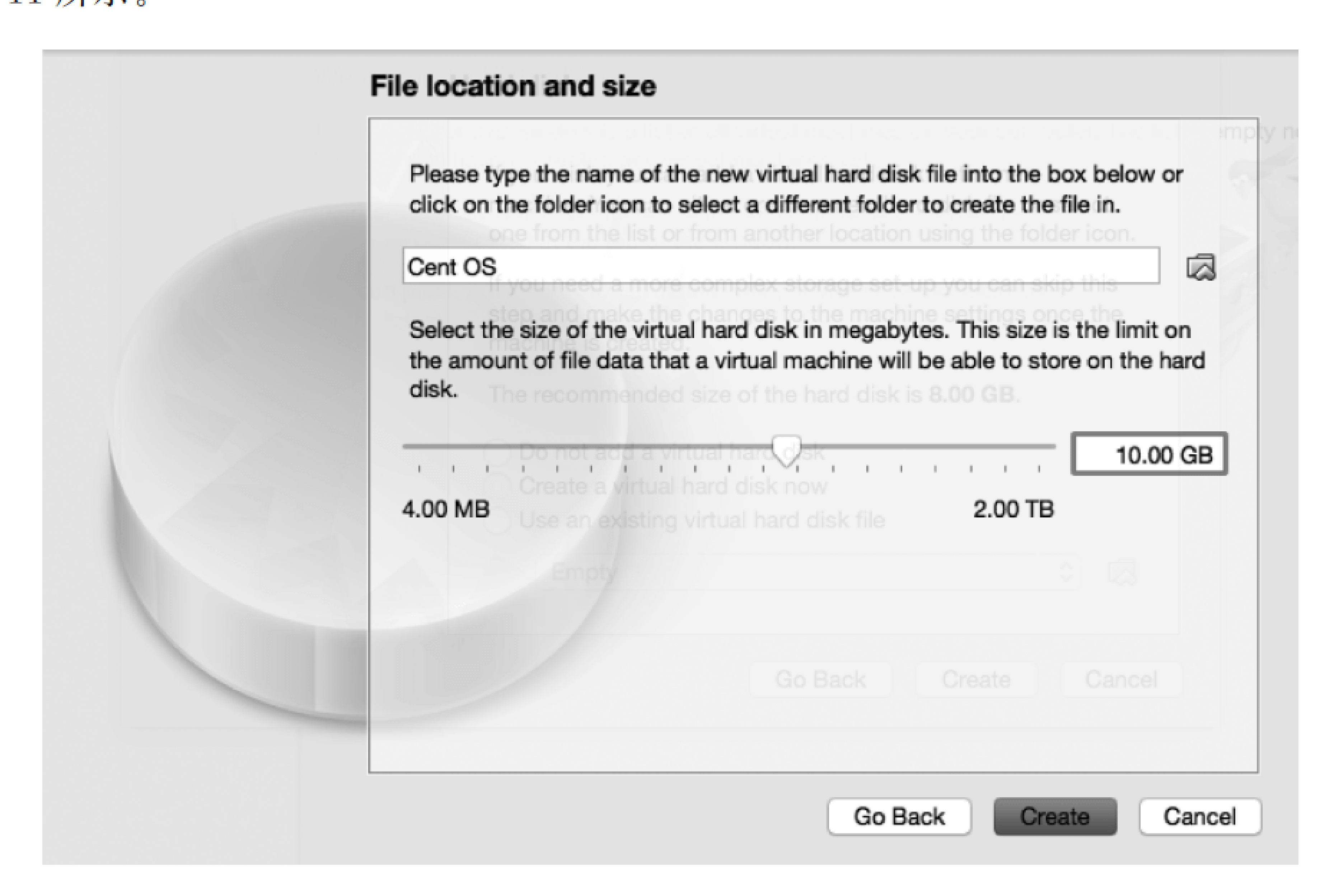


图 7.11 硬盘文件地址设置和文件大小设置

最后单击 Create 按钮完成虚拟机配置。接下来,要启动虚拟机进行 Cent OS 安装,选择 Start 命令,如图 7.12 所示。

启动之后选择下载好的安装 ISO 镜像,选择 Start 命令,如图 7.13 所示。

之后会开始安装过程,出现界面后按 Enter 键,进入安装流程。第一步选择【语言】|【中



图 7.12　虚拟机界面



图 7.13　选择镜像

文】命令，如图 7.14 所示。

单击【继续】按钮，进入综合配置，选择【安装位置】|【自动分区】命令后，单击【完成】按钮，如图 7.15 所示。

选择【软件选择】命令，再选择【GNOME 桌面】命令，单击【完成】按钮，如图 7.16 所示。

之后单击【开始安装】按钮，设置 Root 密码并创建用户，等待，如图 7.17 所示。



图 7.14 语言选择



图 7.15 综合配置



图 7.16　软件选择



图 7.17　配置

安装结束后单击【重启】按钮，即可完成，如图 7.18 所示。

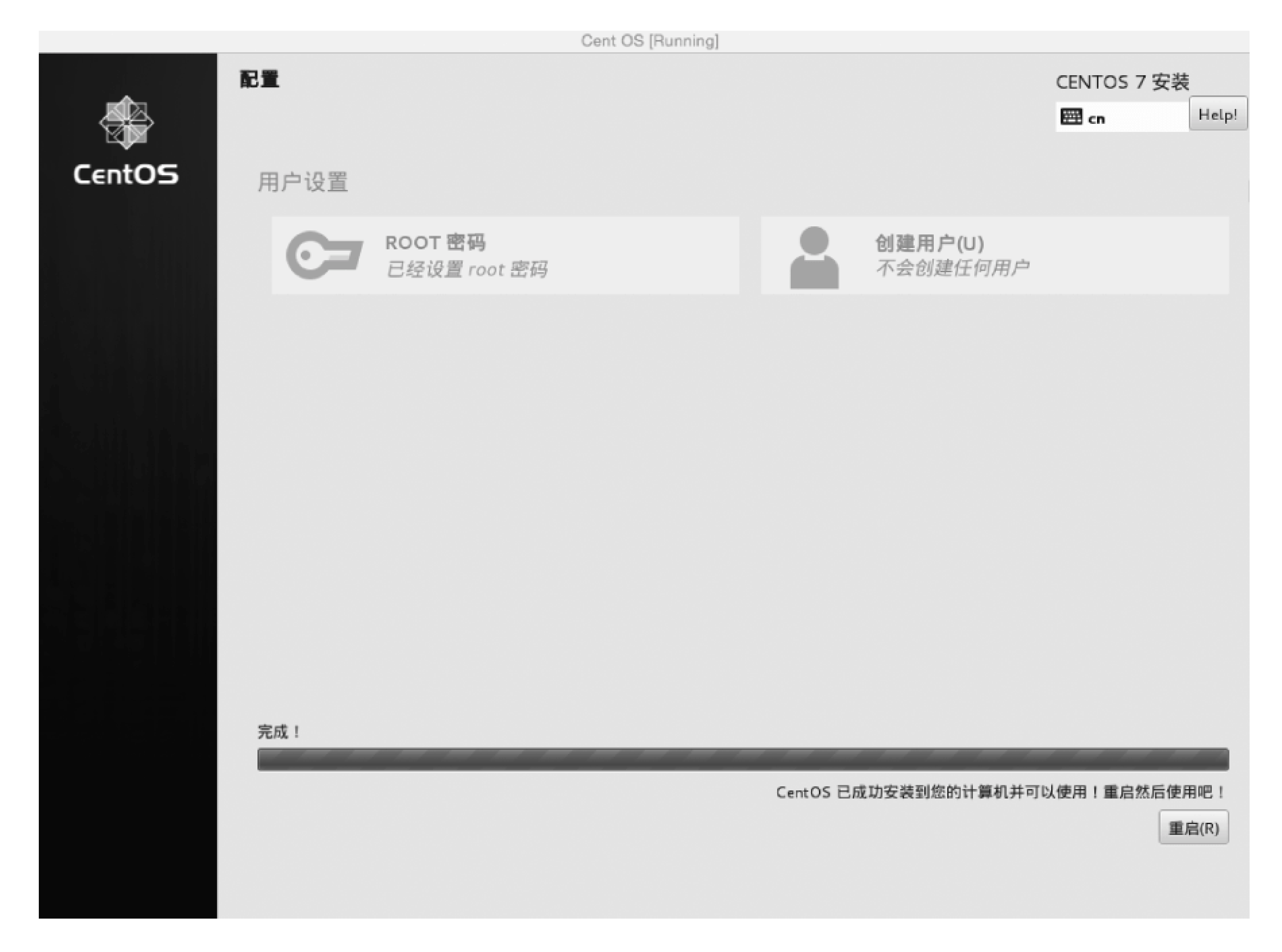


图 7.18　完成安装

重启后完成相关设置即可使用，如图 7.19 所示，单击 Start using CentOS Linux 按钮。

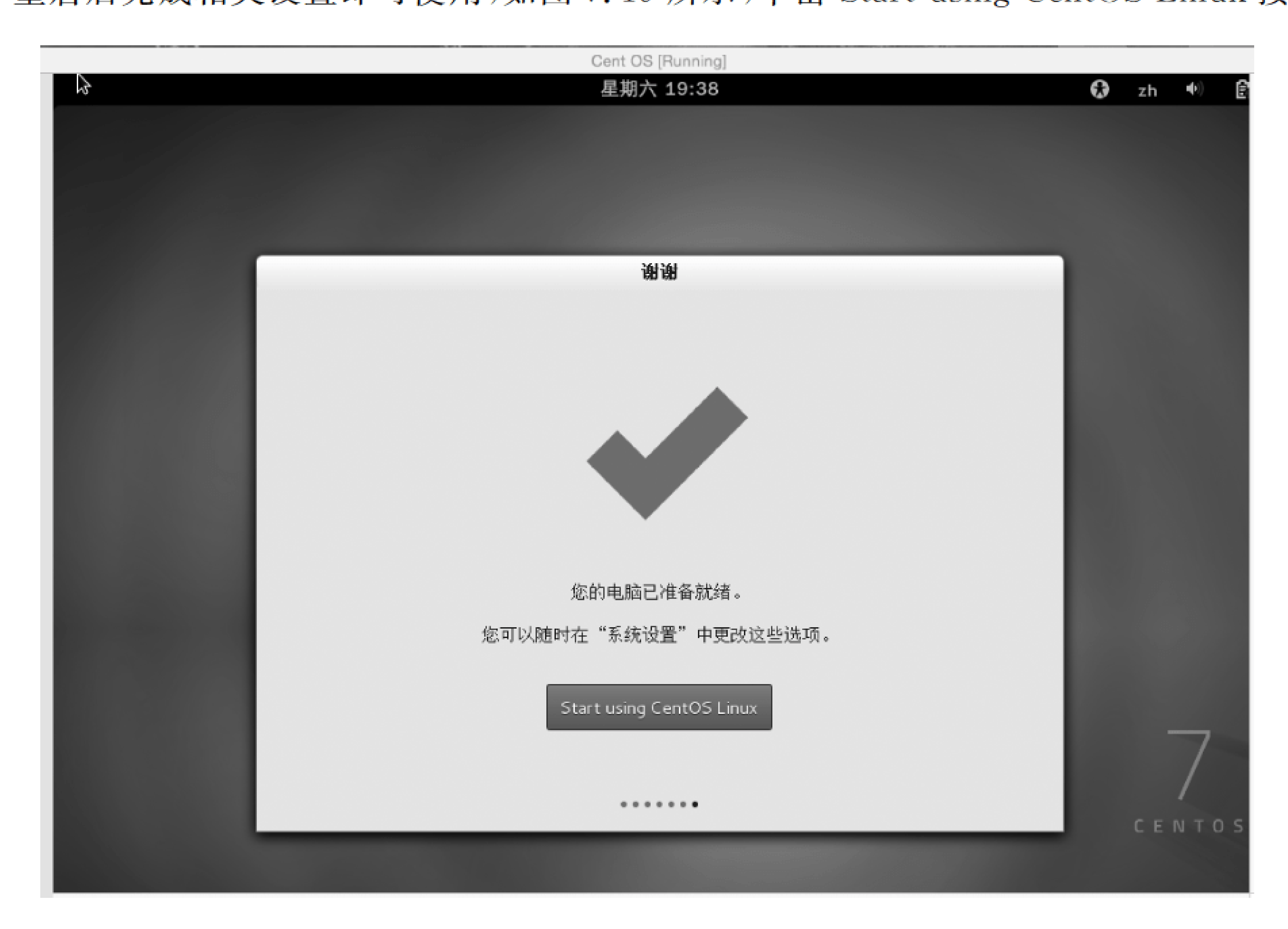


图 7.19　完成配置

最后可以在 Virtual Box【菜单】中,选择安装 Virtual Box Guest Additions 以保证系统的最佳状态。

7.2.2 JDK 安装

由于 Hadoop 是由 Java 语言开发的,所以需要安装 JDK,如图 7.20 所示。



图 7.20　JDK 下载页面

选择 JDK 下面的 Download 选项卡,进入 JDK 下载页面。选择 Accept License Agreement 选项接受条款,选择 rpm 版本的 Linux x64 平台下的 JDK 进行下载,如图 7.21 所示。之后等待即可。

在安装 JDK 之前,需要卸载 Cent OS 自带的 OpenJDK。选择【应用程序】|【终端】命令。

输入命令 rpm-qa|grep openjdk-i,查询已经安装的 OpenJDK,如图 7.22 所示。

输入命令 sudo yum remove java-1.7.0-openjdk-1.7.0.75-2.5.4.2.el7_0.x86_64 java-1.7.0-openjdk-headless-1.7.0.75-2.5.4.2.el7_0.x86_64 卸载 OpenJDK。将会要求输入密码,并会要求确认,输入 y 后按 Enter 键,即可确定完成卸载,如图 7.23 所示。

之后进行安装 JDK,运行下载的 jdk-8u51-linux-x64.rpm 程序,单击【安装】按钮,进行 JDK 的安装,如图 7.24 所示。

之后打开终端,输入 java-version,出现如图 7.25 的信息则说明安装成功。

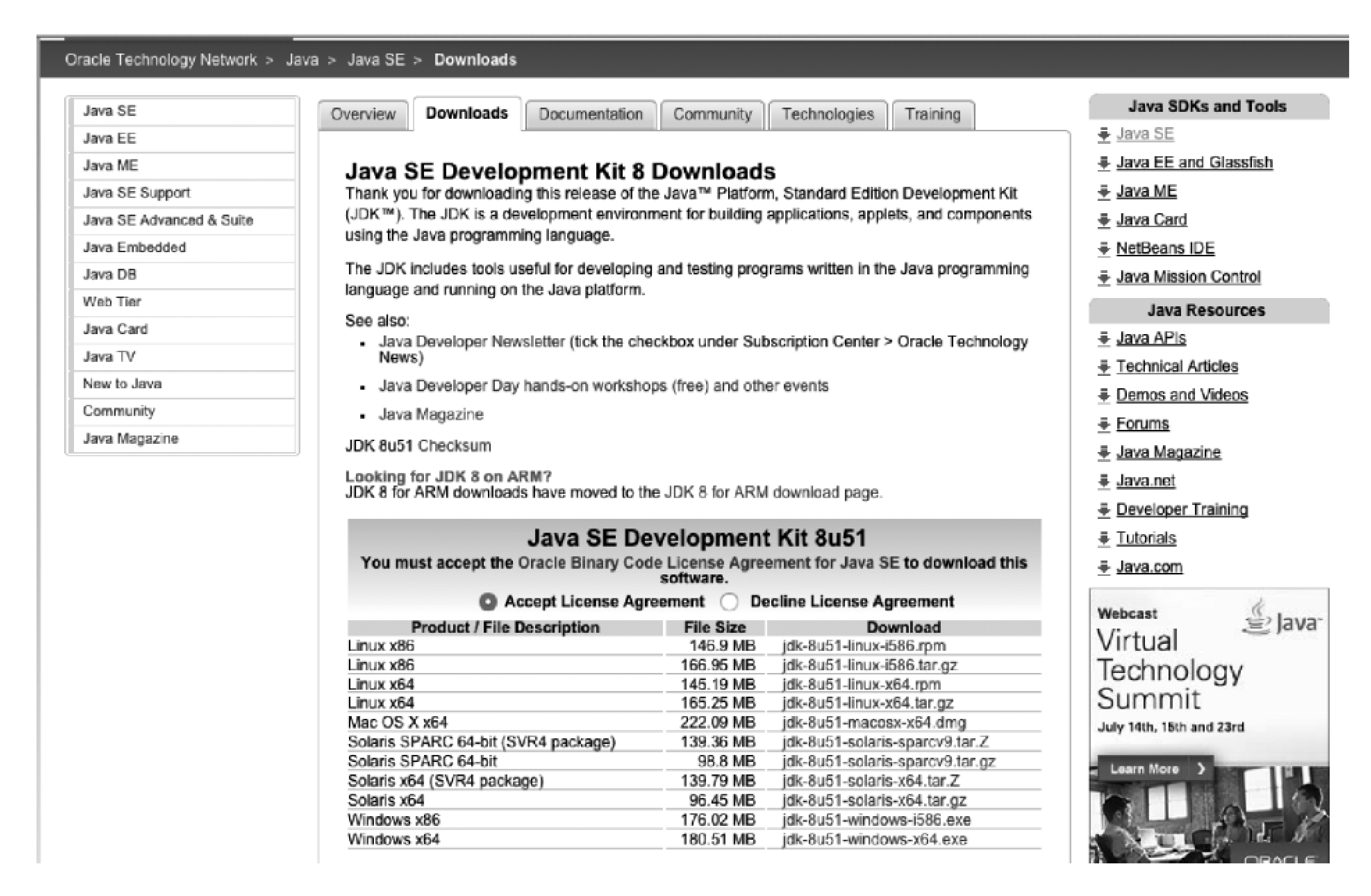


图 7.21 JDK 下载页面

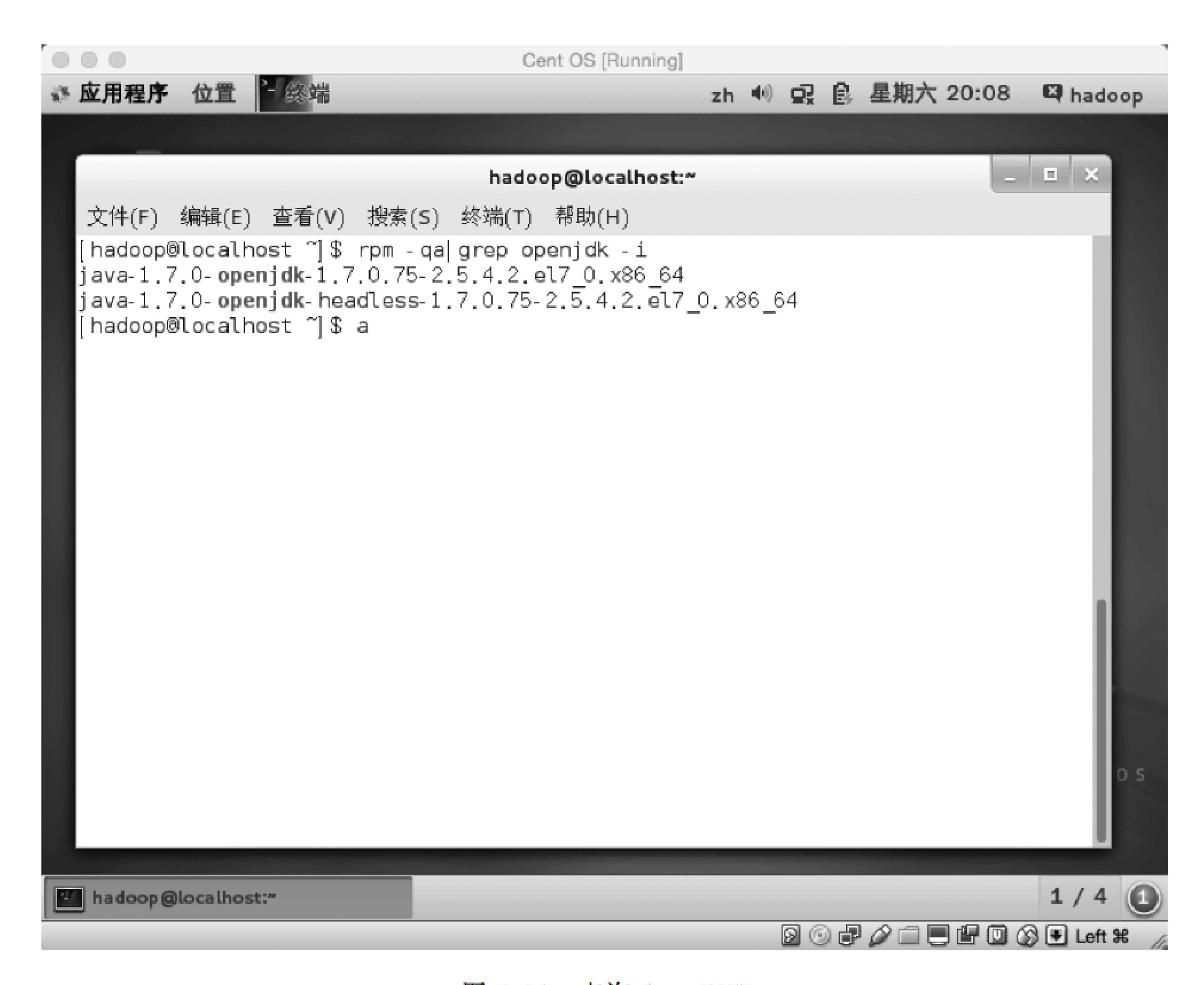


图 7.22 查询 OpenJDK

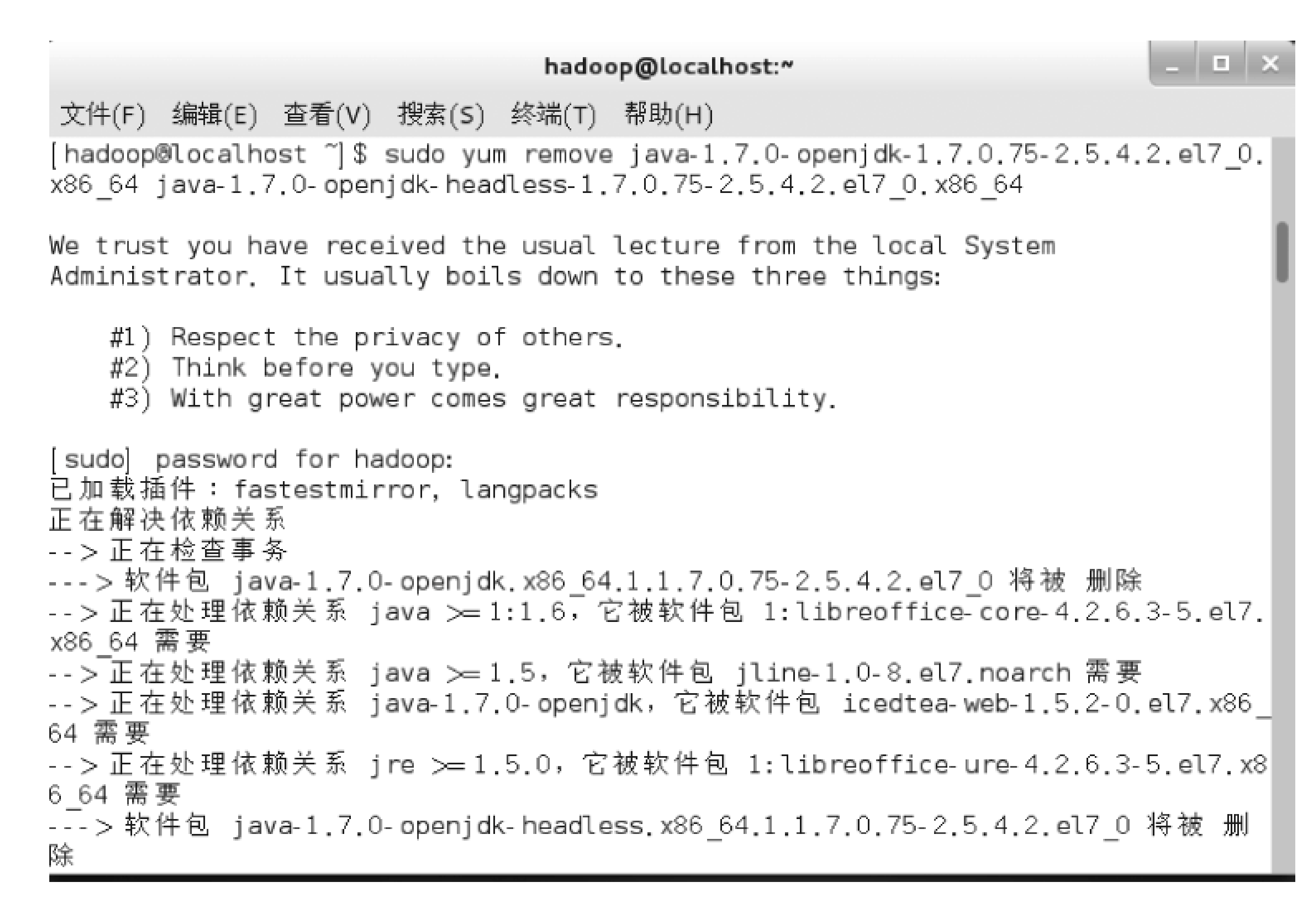


图 7.23 卸载 OpenJDK



图 7.24 JDK 安装

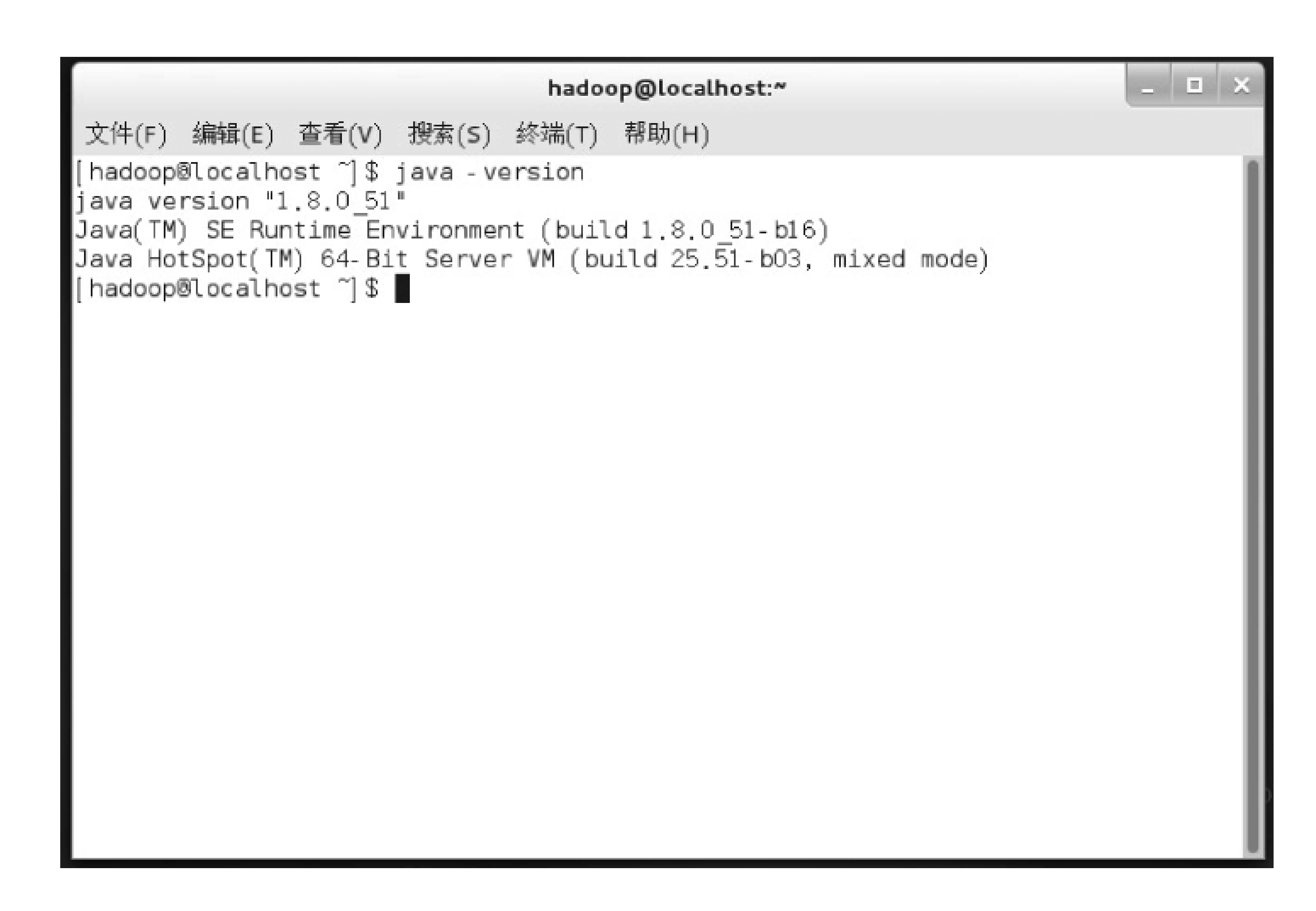


图 7.25 JDK 信息查询

至此,安装 Hadoop 前的全部配置工作已经完成。下面讲解 Hadoop 的安装和配置内容。

7.2.3 Hadoop 下载

通过浏览器访问 Hadoop 官网,地址为 http://hadoop.apache.org/,官方网站如图 7.26 所示。

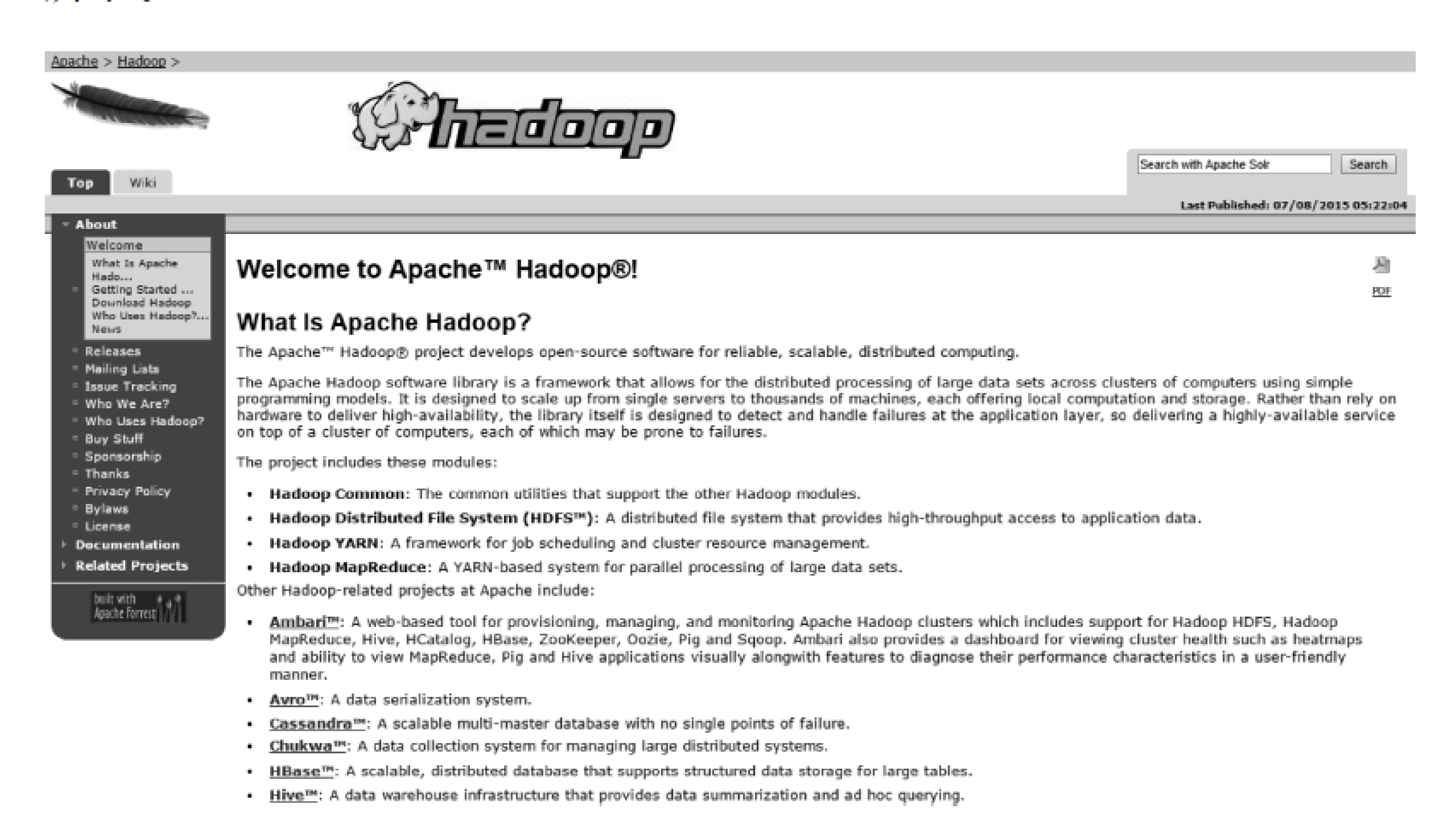


图 7.26 Hadoop 官方网站首页

在右侧边栏中选择 Release 中的 Download 进入 Hadoop 下载信息页面,如图 7.27 所示。

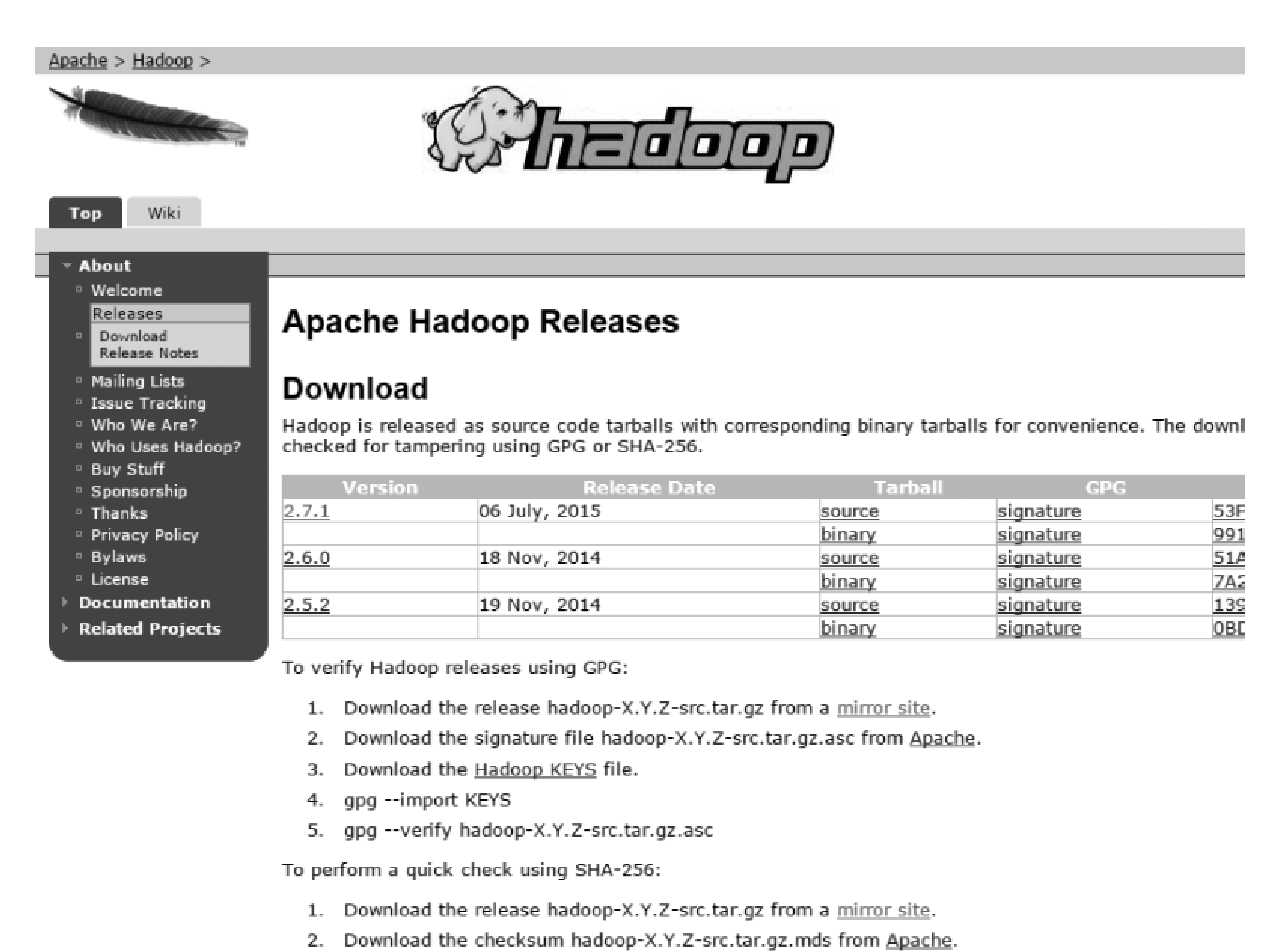


图 7.27 Hadoop 下载页面

在Download项中有可供下载的版本信息、释放日期等信息。每个版本都会提供source和binary版本的下载。source是源代码，在下载后需要自行编译，binary为二进制可执行版本，已经编译完成，下载后即可运行。

选择想要下载的版本的binary，之后进入下载页面。该页面会显示一些下载镜像地址，选择一个开始下载，如图7.28所示。

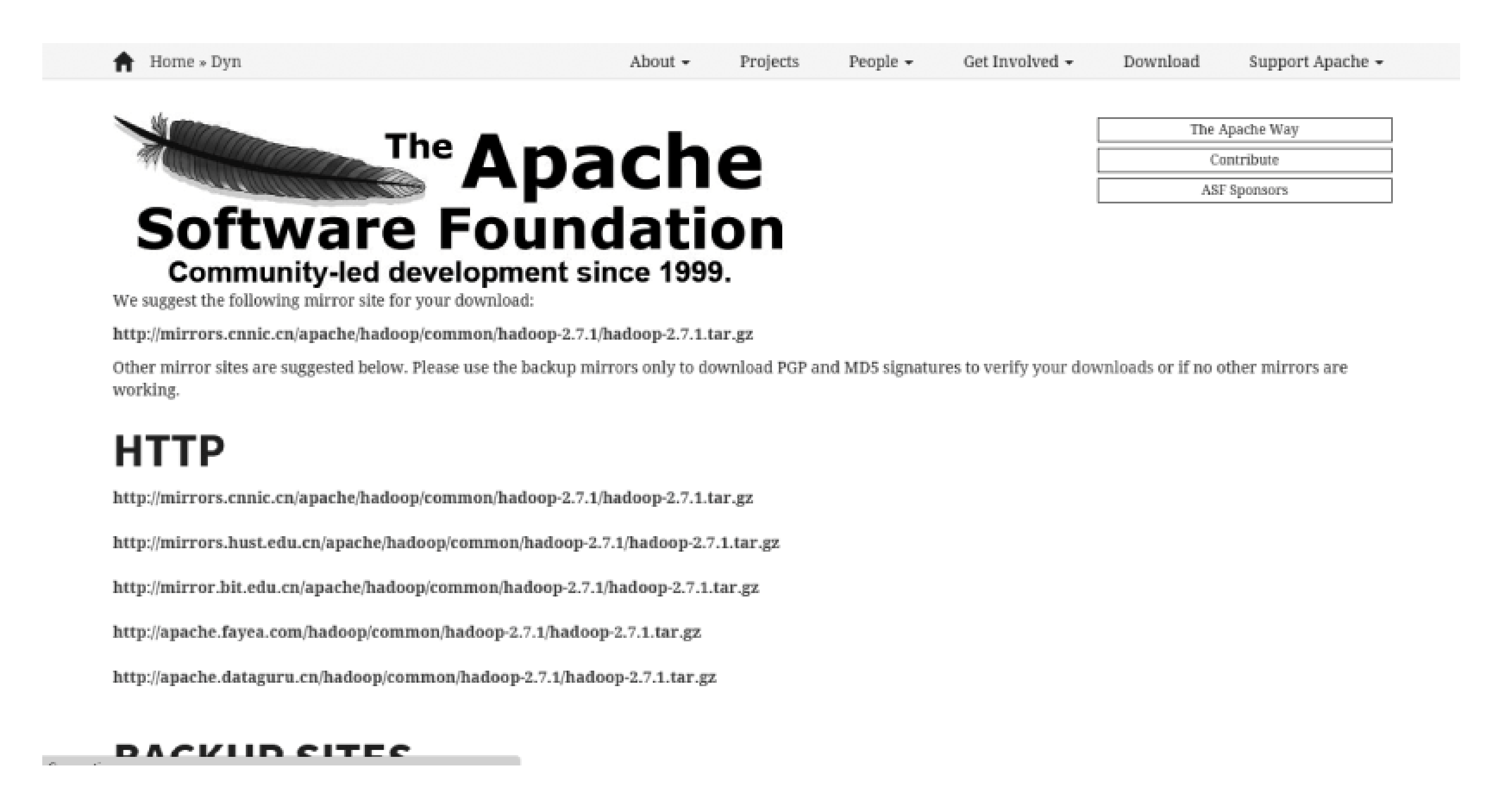


图7.28　Hadoop下载

7.2.4　Hadoop单机模式配置

下载完成后将hadoop-2.7.1.tar.gz解压到桌面。进入hadoop-2.7.1/etc/hadoop文件夹，找到hadoop-env.sh，右击，选择【使用gedit打开】命令，如图7.29所示。

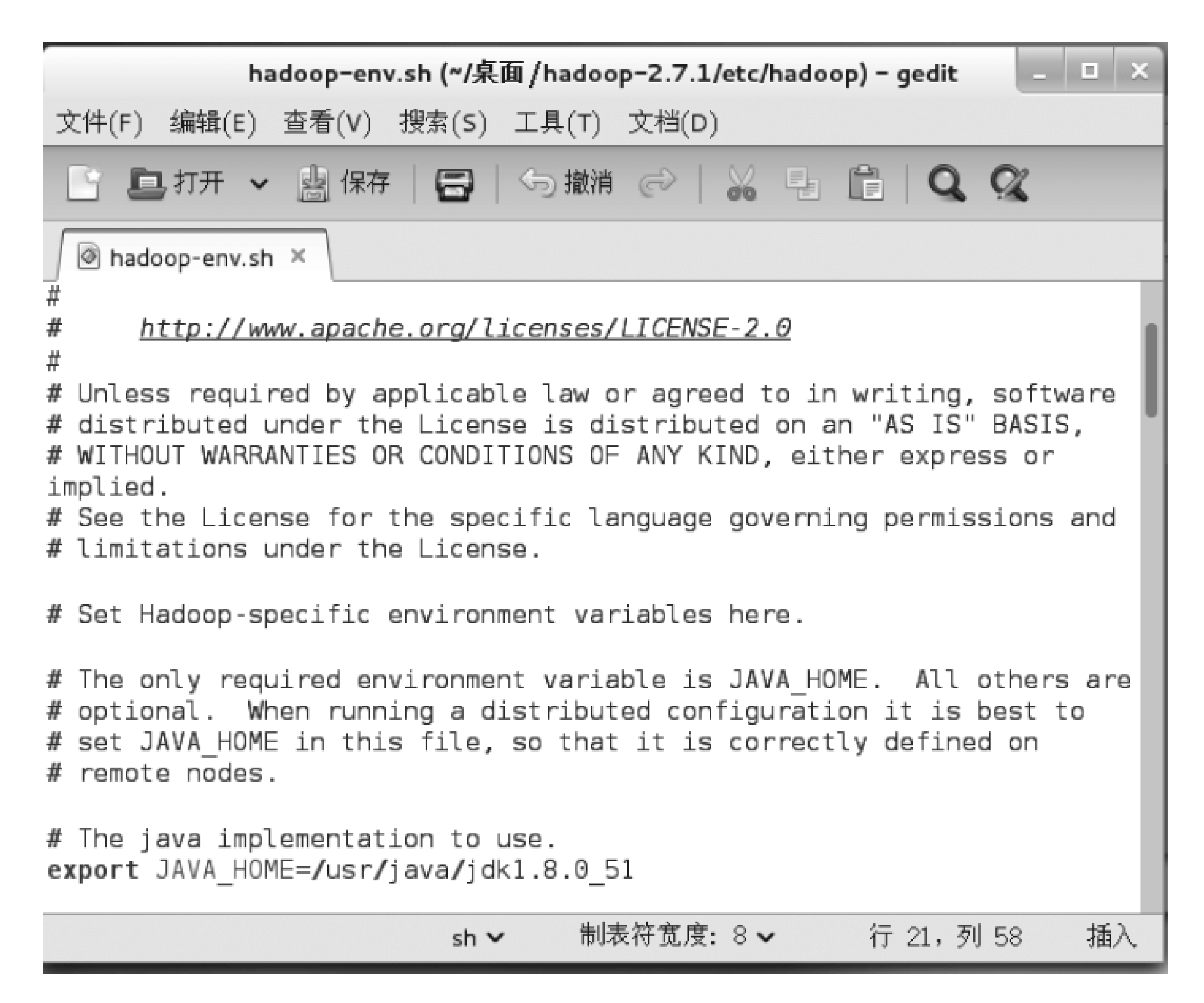


图7.29　hadoop-env.sh文件

修改 export JAVA_HOME=后的内容为/usr/java/jdk1.8.0_51。选择【保存】命令。完成配置。

之后打开终端输入命令

```
cd ~
mkdir input
cp ~/桌面/hadoop-2.7.1/etc/hadoop/*.xml input
~/桌面/hadoop-2.7.1/bin/hadoop jar ~/桌面/hadoop-2.7.1/share/hadoop/mapreduce/
hadoop-mapreduce-examples-2.7.1.jar grep input output 'dfs[a-z.]+'
```

如图 7.30 所示。

```
hadoop@localhost:~
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
[hadoop@localhost ~]$ cd ~
[hadoop@localhost ~]$ mkdir input
[hadoop@localhost ~]$ ls
input 公共 模板 视频 图片 文档 下载 音乐 桌面
[hadoop@localhost ~]$ cp ~/桌面/hadoop-2.7.1/etc/hadoop/*.xml input
[hadoop@localhost ~]$ ~/桌面/hadoop-2.7.1/bin/etc jar ~/桌面/hadoop-2.7.1/s
hare/hadoop/mapreduce/hadoop-mapreduce-examples-2.7.1.jar grep input output
'dfs[a-z.]+'
bash: /home/hadoop/桌面/hadoop-2.7.1/bin/etc: 没有那个文件或目录
[hadoop@localhost ~]$ ~/桌面/hadoop-2.7.1/bin/hadoop jar ~/桌面/hadoop-2.7.
1/share/hadoop/mapreduce/hadoop-mapreduce-examples-2.7.1.jar grep input outp
ut 'dfs[a-z.]+'
15/07/18 23:42:11 INFO Configuration.deprecation: session.id is deprecated.
Instead, use dfs.metrics.session-id
15/07/18 23:42:11 INFO jvm.JvmMetrics: Initializing JVM Metrics with process
Name=JobTracker, sessionId=
15/07/18 23:42:13 INFO input.FileInputFormat: Total input paths to process :
 8
15/07/18 23:42:13 INFO mapreduce.JobSubmitter: number of splits:8
15/07/18 23:42:13 INFO mapreduce.JobSubmitter: Submitting tokens for job: jo
b_local373032523_0001
15/07/18 23:42:14 INFO mapreduce.Job: The url to track the job: http://local
host:8080/
```

图 7.30 Hadoop 单机模式

7.2.5 Hadoop 伪分布模式配置

首先进入 hadoop-2.7.1/etc/hadoop 文件夹，在 core-site.xml 文件上右击，选择【打开方式】|【使用 gedit 打开】命令。

将如下代码填写到文件中，并保存，如图 7.31 所示。

```
<configuration>
    <property>
        <name>fs.defaultFS</name>
        <value>hdfs://localhost:9000</value>
    </property>
</configuration>
```

之后找到 hdfs-site.xml 文件，并使用 gedit 打开，将如下代码填写到文件中，并保存文件，如图 7.32 所示。

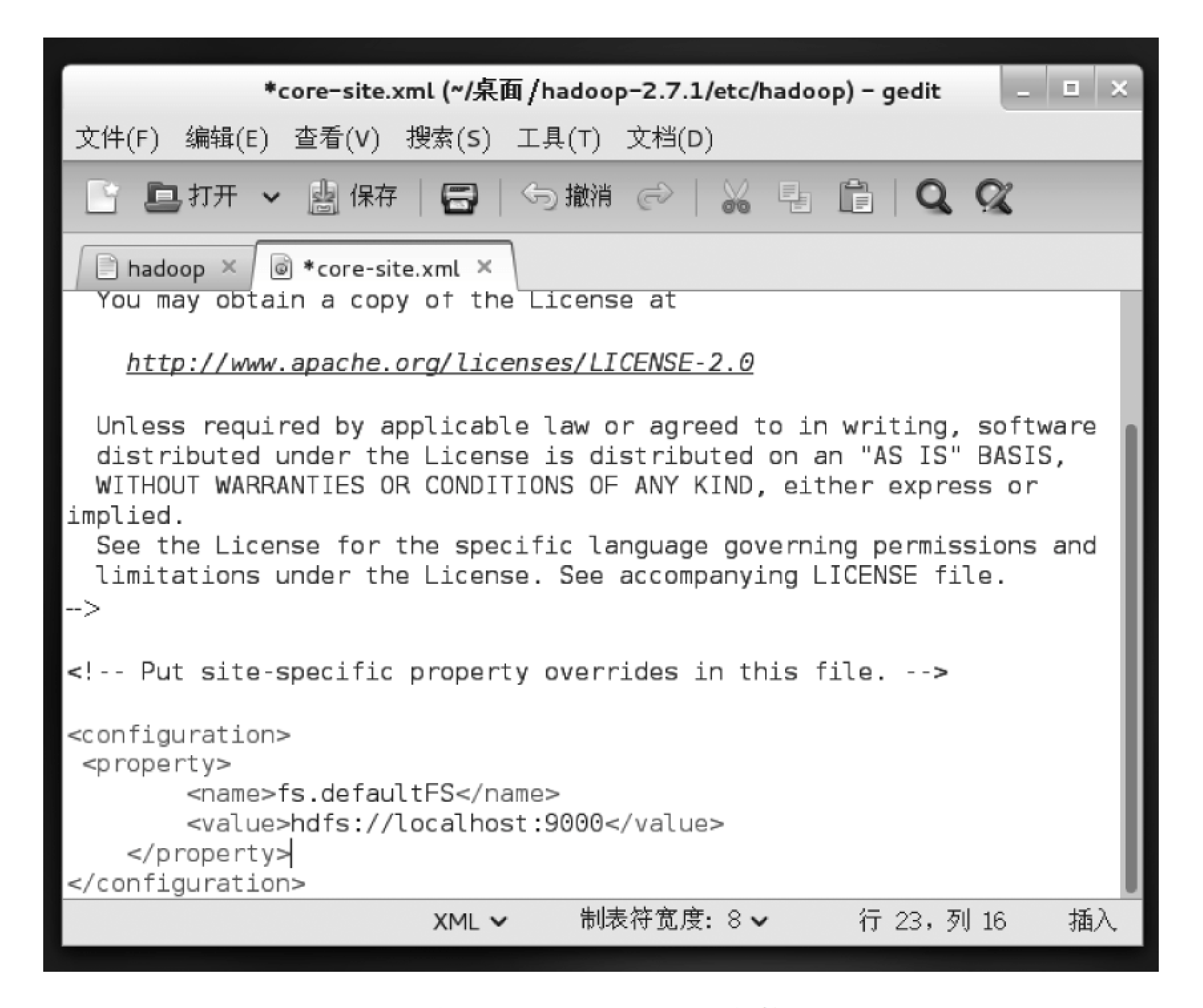


图 7.31　core-site. xml 文件



图 7.32　hdfs-site. xml 文件

```
<configuration>
    <property>
        <name>dfs.replication</name>
        <value>1</value>
    </property>
</configuration>
```

接下来进行免密码登录 SSH 设置，打开终端输入命令 ssh localhost，如图 7.33 所示，如果需要输入密码登录 SSH，则需要输入命令设置免登录 SSH。



图 7.33 SSH 登录测试

输入以下代码进行设置

```
ssh-keygen -t dsa -P '' -f ~/.ssh/id_dsa
cat ~/.ssh/id_dsa.pub >> ~/.ssh/authorized_keys
export HADOOP\_PREFIX=/usr/local/Hadoop
```

如图 7.34 所示。

```
hadoop@localhost:~
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
hadoop@localhost's password:
Last failed login: Sun Jul 19 07:51:34 CST 2015 from localhost on ssh
There were 3 failed login attempts since the last successful login.
Last login: Sat Jul 18 20:40:04 2015
[hadoop@localhost ~]$ ssh-keygen -t dsa -P '' -f ~/.ssh/id_dsa
Generating public/private dsa key pair.
Your identification has been saved in /home/hadoop/.ssh/id_dsa.
Your public key has been saved in /home/hadoop/.ssh/id_dsa.pub.
The key fingerprint is:
28:33:63:48:51:6a:db:b6:ff:5c:f2:8f:8f:b8:eb:c4 hadoop@localhost.loca
The key's randomart image is:
+--[ DSA 1024]----+
|   ...           |
|    o            |
|   +             |
|  o +    .       |
|   o O . S       |
|    o * .        |
|     .   E .     |
|      . o = o    |
|       .oBo+oo   |
+-----------------+
[hadoop@localhost ~]$  cat ~/.ssh/id_dsa.pub >> ~/.ssh/authorized_key
```

图 7.34 SSH 免登录设置

接下来介绍在本地运行 MapReduce。

首先要修改几个配置文件，打开 hadpoop-2.7.1 文件夹，进入 etc 文件夹中 hadoop 文件夹内，找到 yarn-env.sh，使用 gedit 打开它。找到# someJava parameters 这行代码，将其下方的代码去掉#号注释，并修改为 export JAVA_HOME=/usr/java/jdk1.8.0_51。之后将文件保存，如图 7.35 所示。

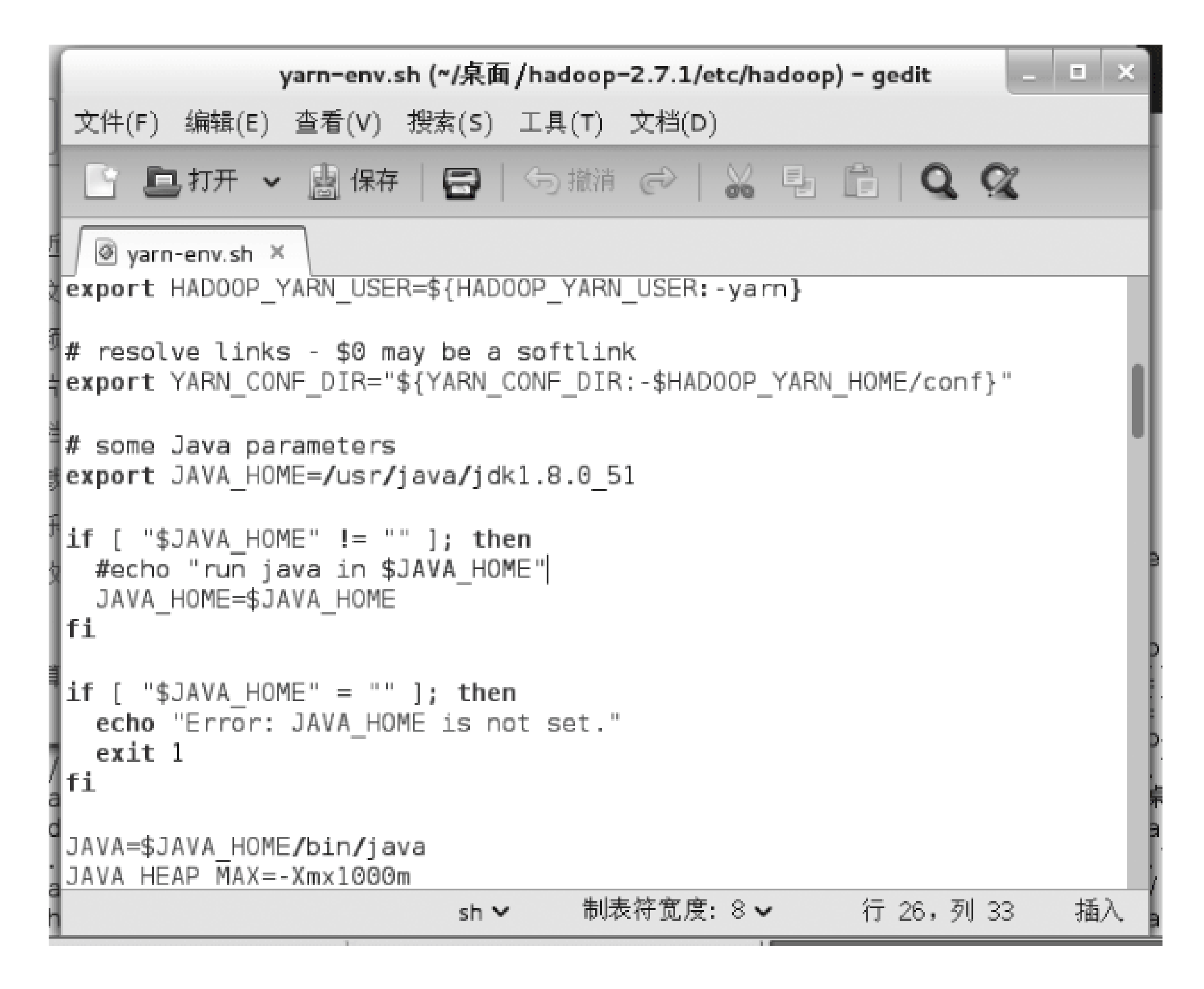


图 7.35 yarn-env.sh 文件设置

接下来进入 hadoop-2.7.1 文件夹下的 libexec 文件夹，找到 hadoop-config.sh 文件，并使用 gedit 打开。

找到如下的代码：

```
# Newer versions of glibc use an arena memory allocator that causes virtual
# memory usage to explode. This interacts badly with the many threads that
# we use in Hadoop. Tune the variable down to prevent vmem explosion.
```

在其下方添加 export JAVA_HOME=/usr/java/jdk1.8.0_51。之后将文件保存，如图 7.36 所示。

首先要格式化 HDFS 文件系统。打开终端，输入如下命令并执行：

```
cd ~/桌面/hadoop-2.7.1/
bin/hdfs namenode - format
```

如果执行后出现错误，重启系统，如图 7.37 所示。

之后启动 NameNode 守护进程和 DataNode 守护进程。在终端中执行如下命令：

```
sbin/start-dfs.sh
```

之后可能会要求输入用户密码，输入正确密码并按 Enter 键，如图 7.38 所示。Hadoop 守护日志会写入 $HADOOP_LOG_DIR 文件夹中(默认在 $HADOOP_HOME/logs 中)。

```
hadoop-config.sh (~/桌面/hadoop-2.7.1/libexec) - gedit
文件(F) 编辑(E) 查看(V) 搜索(S) 工具(T) 文档(D)
打开 保存 撤消
hadoop-config.sh ×
  exit 1
fi

# Newer versions of glibc use an arena memory allocator that causes
virtual
# memory usage to explode. This interacts badly with the many threads
that
# we use in Hadoop. Tune the variable down to prevent vmem explosion.
export MALLOC_ARENA_MAX=${MALLOC_ARENA_MAX:-4}
export JAVA_HOME=/usr/java/jdk1.8.0_51
# Attempt to set JAVA_HOME if it is not set
if [[ -z $JAVA_HOME ]]; then
  # On OSX use java_home (or /Library for older versions)
  if [ "Darwin" == "$(uname -s)" ]; then
    if [ -x /usr/libexec/java_home ]; then
      export JAVA_HOME=($(/usr/libexec/java_home))
    else
      export JAVA_HOME=(/Library/Java/Home)
    fi
  fi
sh  制表符宽度: 8  行 162, 列 76  插入
```

图 7.36 hadoop-config.sh 文件设置

```
hadoop@localhost:~/桌面/hadoop-2.7.1
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
[hadoop@localhost ~]$ cd ~/桌面/hadoop-2.7.1/
[hadoop@localhost hadoop-2.7.1]$ bin/hdfs namenode -format
15/07/21 22:05:18 INFO namenode.NameNode: STARTUP_MSG:
/************************************************************
STARTUP_MSG: Starting NameNode
STARTUP_MSG:   host = localhost/127.0.0.1
STARTUP_MSG:   args = [-format]
STARTUP_MSG:   version = 2.7.1
STARTUP_MSG:   classpath = /home/hadoop/桌面/hadoop-2.7.1/etc/hadoop:/home/hadoo
p/桌面/hadoop-2.7.1/share/hadoop/common/lib/commons-configuration-1.6.jar:/home/
hadoop/桌面/hadoop-2.7.1/share/hadoop/common/lib/curator-client-2.7.1.jar:/home/
hadoop/桌面/hadoop-2.7.1/share/hadoop/common/lib/gson-2.2.4.jar:/home/hadoop/桌
面/hadoop-2.7.1/share/hadoop/common/lib/activation-1.1.jar:/home/hadoop/桌面/had
oop-2.7.1/share/hadoop/common/lib/jackson-jaxrs-1.9.13.jar:/home/hadoop/桌面/had
oop-2.7.1/share/hadoop/common/lib/jsp-api-2.1.jar:/home/hadoop/桌面/hadoop-2.7.1
/share/hadoop/common/lib/jaxb-impl-2.2.3-1.jar:/home/hadoop/桌面/hadoop-2.7.1/sh
are/hadoop/common/lib/apacheds-kerberos-codec-2.0.0-M15.jar:/home/hadoop/桌面/ha
doop-2.7.1/share/hadoop/common/lib/commons-io-2.4.jar:/home/hadoop/桌面/hadoop-2
.7.1/share/hadoop/common/lib/paranamer-2.3.jar:/home/hadoop/桌面/hadoop-2.7.1/sh
are/hadoop/common/lib/httpclient-4.2.5.jar:/home/hadoop/桌面/hadoop-2.7.1/share/
hadoop/common/lib/log4j-1.2.17.jar:/home/hadoop/桌面/hadoop-2.7.1/share/hadoop/c
ommon/lib/htrace-core-3.1.0-incubating.jar:/home/hadoop/桌面/hadoop-2.7.1/share/
hadoop/common/lib/jets3t-0.9.0.jar:/home/hadoop/桌面/hadoop-2.7.1/share/hadoop/c
ommon/lib/zookeeper-3.4.6.jar:/home/hadoop/桌面/hadoop-2.7.1/share/hadoop/common
```

图 7.37 格式化 HDFS 文件系统

```
hadoop@localhost:~/桌面/hadoop-2.7.1
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
15/07/21 22:05:21 INFO util.GSet: 0.029999999329447746% max memory
7.0 KB
15/07/21 22:05:21 INFO util.GSet: capacity      = 2^15 = 32768 entr
15/07/21 22:05:21 INFO namenode.FSImage: Allocated new BlockPoolId:
-127.0.0.1-1437487521186
15/07/21 22:05:21 INFO common.Storage: Storage directory /tmp/hadoo
name has been successfully formatted.
15/07/21 22:05:21 INFO namenode.NNStorageRetentionManager: Going to
ges with txid >= 0
15/07/21 22:05:21 INFO util.ExitUtil: Exiting with status 0
15/07/21 22:05:21 INFO namenode.NameNode: SHUTDOWN_MSG:
/************************************************************
SHUTDOWN_MSG: Shutting down NameNode at localhost/127.0.0.1
************************************************************/
[hadoop@localhost hadoop-2.7.1]$ sbin/start-dfs.sh
Starting namenodes on [localhost]
hadoop@localhost's password:
localhost: starting namenode, logging to /home/hadoop/桌面/hadoop-2
oop-hadoop-namenode-localhost.localdomain.out
hadoop@localhost's password:
```

图 7.38 启动 Hadoop 守护进程

使用浏览器访问 NameNode 网页界面的默认网址 http://localhost:50070/，如图 7.39 所示。

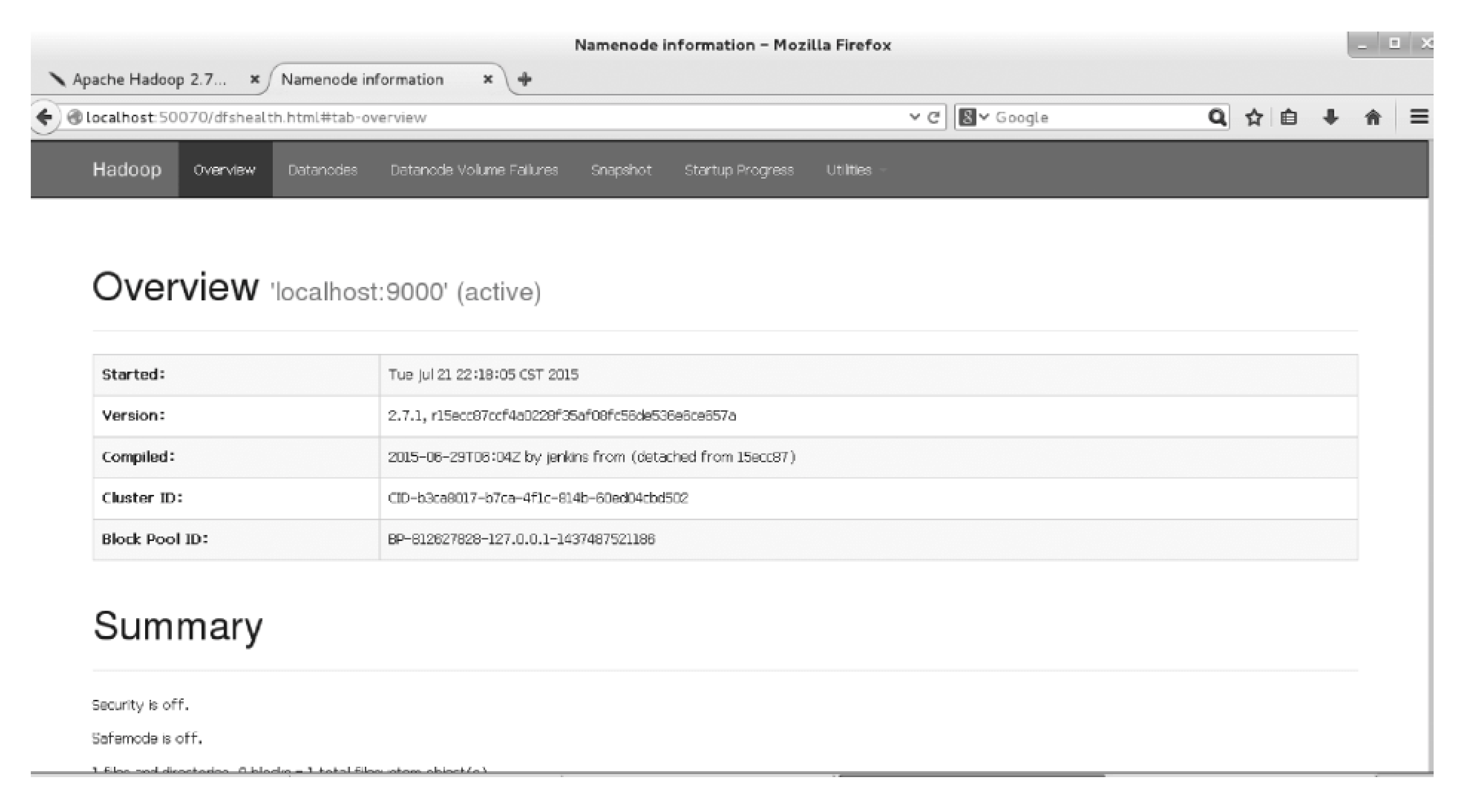


图 7.39　NameNode 网页界面

使 HDFS 文件夹被要求处理 MapReduce 的工作，在终端输入如下代码：

```
bin/hdfs dfs -mkdir /user
bin/hdfs dfs -mkdir /user/<username>
```

如图 7.40 所示。

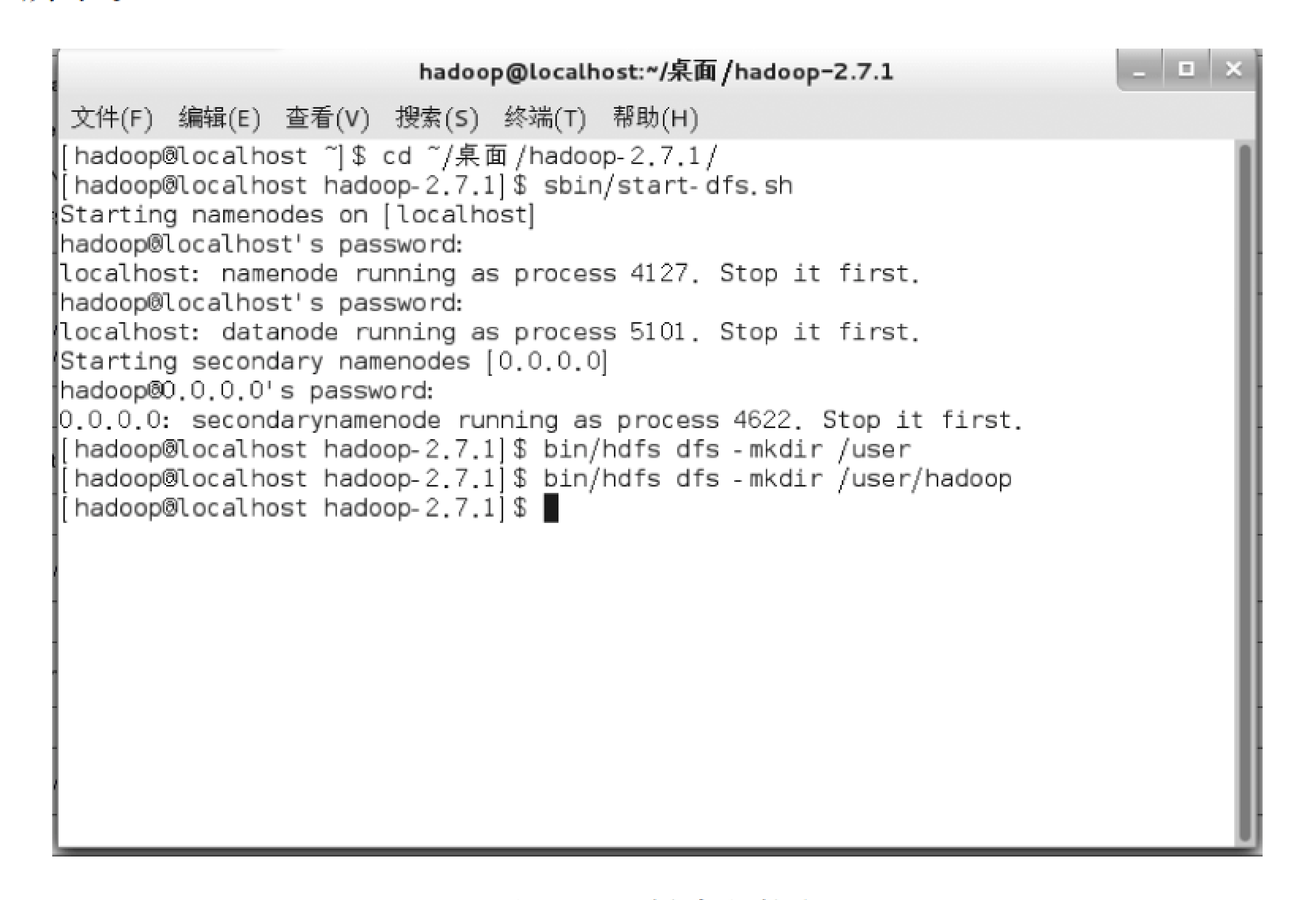


图 7.40　创建文件夹

接下来将输入文件复制到分布式文件系统中，在终端输入如下代码：

```
bin/hdfs dfs -put etc/hadoop input
```

如图 7.41 所示。

```
hadoop@localhost:~/桌面/hadoop-2.7.1
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
[hadoop@localhost hadoop-2.7.1]$ bin/hdfs dfs -put etc/hadoop input

[hadoop@localhost hadoop-2.7.1]$
[hadoop@localhost hadoop-2.7.1]$
```

图 7.41 输入文件复制到分布式文件系统

接下来运行提供的一些样例，在终端输入如下代码：

```
bin/hadoop jar share/hadoop/mapreduce/hadoop - mapreduce - examples - 2.7.1.jar grep input
output 'dfs[a - z.] + '
```

如图 7.42 所示。

```
hadoop@localhost:~/桌面/hadoop-2.7.1
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
                Combine input records=0
                Combine output records=0
                Reduce input groups=5
                Reduce shuffle bytes=291
                Reduce input records=11
                Reduce output records=11
                Spilled Records=22
                Shuffled Maps =1
                Failed Shuffles=0
                Merged Map outputs=1
                GC time elapsed (ms)=50
                Total committed heap usage (bytes)=294395904
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
        File Input Format Counters
                Bytes Read=437
        File Output Format Counters
                Bytes Written=197
[hadoop@localhost hadoop-2.7.1]$
```

图 7.42 运行样例代码

接下来查看样例输出，将样例输出从分布式文件系统复制到本地，并进行查看，在终端输入如下代码：

```
bin/hdfs dfs -get output output
cat output/*
```

如图 7.43 所示。

```
hadoop@localhost:~/桌面/hadoop-2.7.1
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
        File Input Format Counters
                Bytes Read=437
        File Output Format Counters
                Bytes Written=197
[hadoop@localhost hadoop-2.7.1]$ bin/hdfs dfs -get output output
15/07/21 22:54:01 WARN hdfs.DFSClient: DFSInputStream has been closed already
15/07/21 22:54:01 WARN hdfs.DFSClient: DFSInputStream has been closed already
[hadoop@localhost hadoop-2.7.1]$ cat output/*
8       dfs.audit.logger
4       dfs.class
3       dfs.server.namenode.
2       dfs.period
2       dfs.audit.log.maxfilesize
2       dfs.audit.log.maxbackupindex
1       dfsmetrics.log
1       dfsadmin
1       dfs.servers
1       dfs.replication
1       dfs.file
[hadoop@localhost hadoop-2.7.1]$
```

图 7.43 在本地查看样例输出

也可以在分布式文件系统中直接查看样例输出信息，在终端输入如下代码：

```
bin/hdfs dfs －cat output/＊
```

如图 7.44 所示。

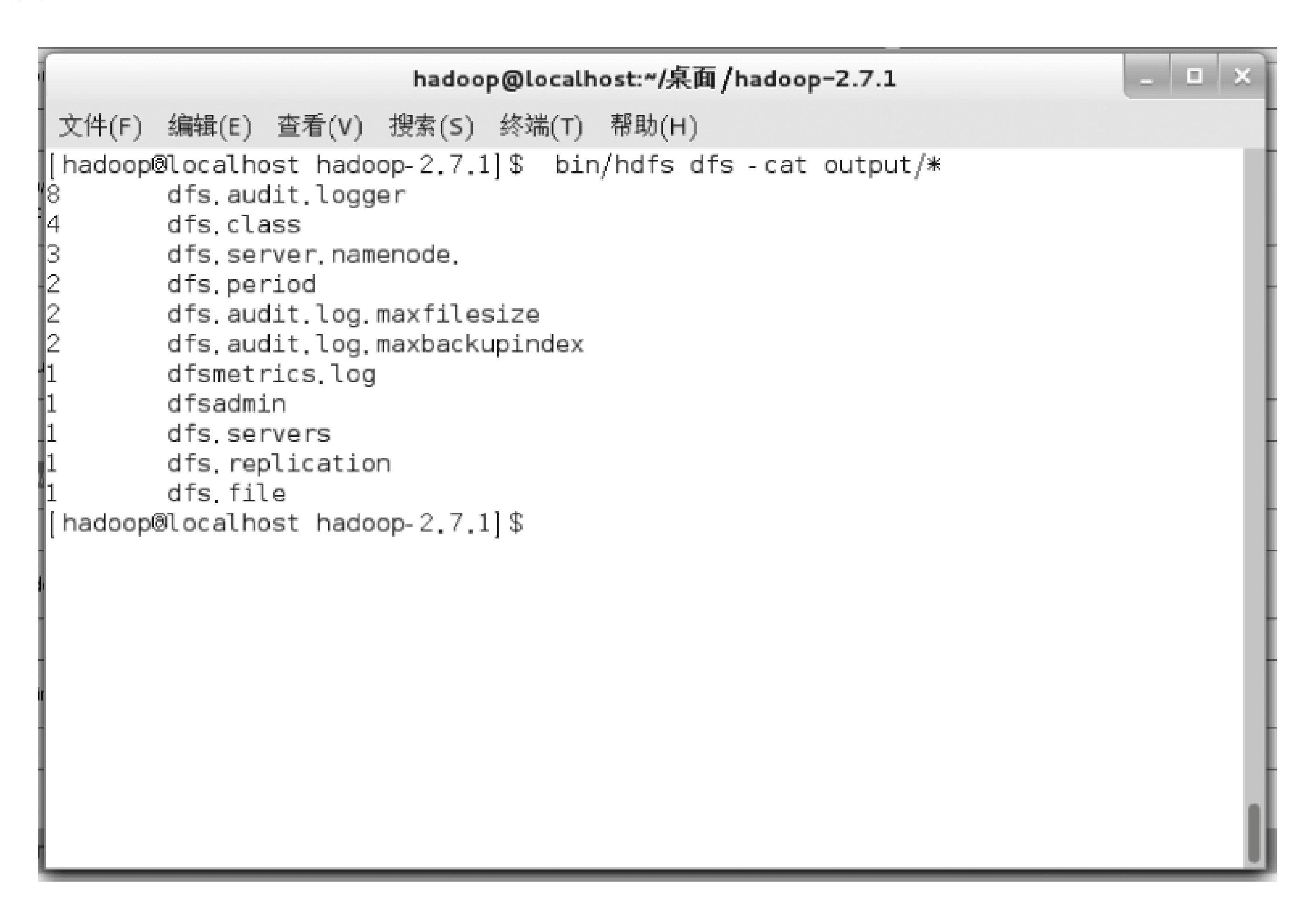


图 7.44 在分布式文件系统中查看样例输出

进行完所有的工作之后，可以将守护进程终止。在终端输入如下代码：

```
sbin/stop－dfs.sh
```

该过程会要求输入密码，输入正确的用户密码，如图 7.45 所示。

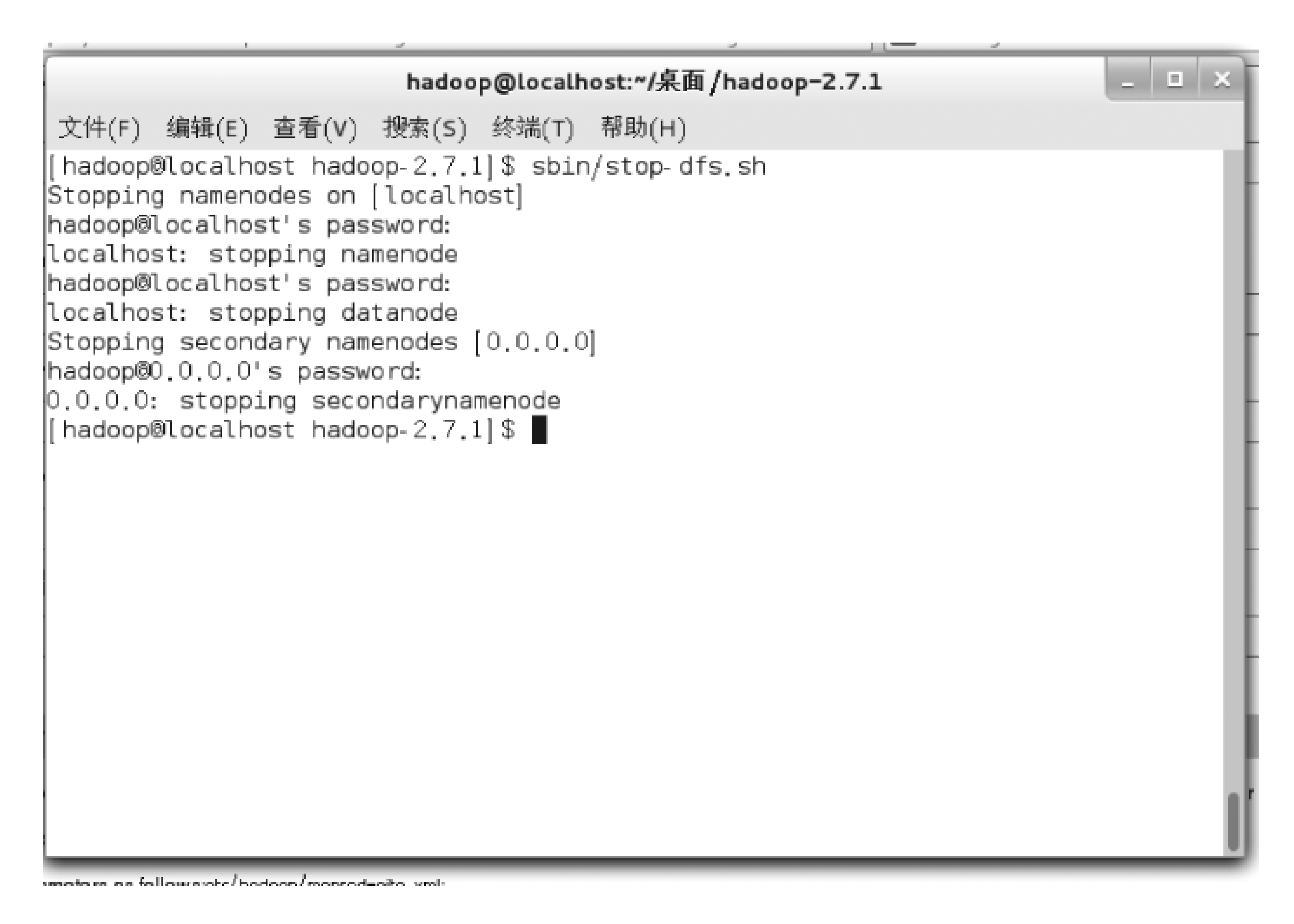

hadoop@localhost:~/桌面/hadoop-2.7.1
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
[hadoop@localhost hadoop-2.7.1]$ sbin/stop-dfs.sh
Stopping namenodes on [localhost]
hadoop@localhost's password:
localhost: stopping namenode
hadoop@localhost's password:
localhost: stopping datanode
Stopping secondary namenodes [0.0.0.0]
hadoop@0.0.0.0's password:
0.0.0.0: stopping secondarynamenode
[hadoop@localhost hadoop-2.7.1]$

图 7.45　终止守护进程



附录A

本书的一些章节最初以文章的形式发表在 IBM 的 developerWorks 中国网站（ibm. com/developerworks/cn/）上。

developerWorks 中国网站作为 IBM 全球开发者策略的一个重要环节，于 2001 年 2 月发布，是 IBM 面向中国开发者的重要窗口。它面向 IT 行业，提供了专注于 IBM 软件产品和开放标准技术的技术内容、评估软件和社区讨论的最全面的技术资源。网站提供的庞大的、容易搜索的文章和教程文档库（ibm. com/developerworks/cn/library/），可帮助您快速掌握有助您职业发展的重要技能。

developerWorks 可让您轻松地构建帮助您解决问题和获得职业提升的职业关系网。借助 developerWorks 社区（ibm. com/developerworks/mydeveloperworks/connect），您能够：

- 加入这个包含全球数百万 IT 专业人员的网络，寻找您想要认识的同行和专家。
- 通过 developerWorks 群组、博客和论坛进行讨论与协作，获取理想解决方案，从而解决您与同行所面临的棘手技术挑战。

developerWorks 可帮助您在全球同行网络中得到认可。

- 使用 developerWorks 的入门资源，增强您在快速变化的 IT 竞争领域中所需的技能。
- 参与围绕 IBM 软件和开放技术标准的大量论坛，展示您的丰富经验和知识。
- 使用 developerWorks 博客表达您的见解和观点。
- 了解技术社区中的非 IBM 的思想领袖，也称为 IBM Champions。发现、联系或推荐 IBM Champion（ibm. com/developerworks/champion/），了解他们如何最有效地使用 IBM 软件、解决方案和服务。
- 在 developerWorks 上发表技术内容，让您的发布成果获得正式承认。

有关 developerWorks 的更多信息，请访问 ibm. com/developerworks/cn/ 并开始探索。

参 考 文 献

［1］ http://www.ibm.com/developerworks/cn/.

［2］ http://www.cssn.cn/xwcbx/xwcbx_gcsy/201411/t20141104_1388658_6.shtml.

［3］ 王蒙蒙.基于数据消冗技术的大数据加密算法研究.华北水利水电大学硕士学位论文,2013.